Defense Threat Reduction Agency
8725 John J. Kingman Road, MS
6201 Fort Belvoir, VA 22060-6201

DTRA-TR-16-80

TECHNICAL REPORT

Mathematical Approaches to WMD Defense and Vulnerability Assessments of Dynamic Networks

July 2016

HDTRA1-10-1-0050

J. Cole Smith et al.

Prepared by:
University of Florida
339 Weil Hall
Gainesville, FL 32611

DESTRUCTION NOTICE:

Destroy this report when it is no longer needed.
Do not return to sender.

PLEASE NOTIFY THE DEFENSE THREAT REDUCTION AGENCY, ATTN: DTRIAC/ J9STT, 8725 JOHN J. KINGMAN ROAD, MS-6201, FT BELVOIR, VA 22060-6201, IF YOUR ADDRESS IS INCORRECT, IF YOU WISH IT DELETED FROM THE DISTRIBUTION LIST, OR IF THE ADDRESSEE IS NO LONGER EMPLOYED BY YOUR ORGANIZATION.

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704-0188

1. REPORT DATE *(DD-MM-YYYY)*	2. REPORT TYPE	3. DATES COVERED *(From - To)*
00-07-2016	Final	3/30/10 - 8/11/15

4. TITLE AND SUBTITLE
Mathematical Approaches to WMD Defense and Vulnerability Assessments on Dynamic Networks

5a. CONTRACT NUMBER

5b. GRANT NUMBER
HDTRA1-10-01-0050

5c. PROGRAM ELEMENT NUMBER

6. AUTHOR(S)
J. Cole Smith
Panos M. Pardalos
My T. Thai

5d. PROJECT NUMBER

5e. TASK NUMBER

5f. WORK UNIT NUMBER

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)
UNIVERSITY OF FLORIDA
ROSELYN HEATH
339 WEIL HALL
GAINESVILLE FL 32611-6550

8. PERFORMING ORGANIZATION REPORT NUMBER

9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)
Defense Threat Reduction Agency
8725 John J. Kingman Road
Fort Belvoir, VA 22060-6201

10. SPONSOR/MONITOR'S ACRONYM(S)
DTRA

11. SPONSOR/MONITOR'S REPORT NUMBER(S)
DTRA-TR-16-80

12. DISTRIBUTION/AVAILABILITY STATEMENT
Distribution Statement A. Approved for public release; distribution is unlimited.

13. SUPPLEMENTARY NOTES

14. ABSTRACT
This report addresses the study of Network Adaptability from Weapons of Mass Destruction (WMD) Disruption and Cascading Failures. The research studies the problems of combating disruptions due to WMD (C-WMD) on networks, accounting for real-world dynamics on critical infrastructures that are vulnerable to such attacks. Our research examines new modes of attack that are relevant to WMD analysis, and considers networks that are overlays of several individually functioning (but related) structures such as telecommunications, transportation, and electricity grids.

15. SUBJECT TERMS
Cascading failures; network interdiction; autonomous agents; combinatorial optimization; approximation algorithms; defense optimization

16. SECURITY CLASSIFICATION OF:			17. LIMITATION OF ABSTRACT	18. NUMBER OF PAGES	19a. NAME OF RESPONSIBLE PERSON
a. REPORT	b. ABSTRACT	c. THIS PAGE			Peter Vandeventer
U	SAR	SAR	U		19b. TELEPHONE NUMBER *(Include area code)* 703-767-4663

UNIT CONVERSION TABLE

U.S. customary units to and from international units of measurement*

U.S. Customary Units	Multiply by → / ← Divide by†		International Units
Length/Area/Volume			
inch (in)	2.54	$\times 10^{-2}$	meter (m)
foot (ft)	3.048	$\times 10^{-1}$	meter (m)
yard (yd)	9.144	$\times 10^{-1}$	meter (m)
mile (mi, international)	1.609 344	$\times 10^{3}$	meter (m)
mile (nmi, nautical, U.S.)	1.852	$\times 10^{3}$	meter (m)
barn (b)	1	$\times 10^{-28}$	square meter (m^2)
gallon (gal, U.S. liquid)	3.785 412	$\times 10^{-3}$	cubic meter (m^3)
cubic foot (ft^3)	2.831 685	$\times 10^{-2}$	cubic meter (m^3)
Mass/Density			
pound (lb)	4.535 924	$\times 10^{-1}$	kilogram (kg)
unified atomic mass unit (amu)	1.660 539	$\times 10^{-27}$	kilogram (kg)
pound-mass per cubic foot (lb ft^{-3})	1.601 846	$\times 10^{1}$	kilogram per cubic meter (kg m^{-3})
pound-force (lbf avoirdupois)	4.448 222		newton (N)
Energy/Work/Power			
electron volt (eV)	1.602 177	$\times 10^{-19}$	joule (J)
erg	1	$\times 10^{-7}$	joule (J)
kiloton (kt) (TNT equivalent)	4.184	$\times 10^{12}$	joule (J)
British thermal unit (Btu) (thermochemical)	1.054 350	$\times 10^{3}$	joule (J)
foot-pound-force (ft lbf)	1.355 818		joule (J)
calorie (cal) (thermochemical)	4.184		joule (J)
Pressure			
atmosphere (atm)	1.013 250	$\times 10^{5}$	pascal (Pa)
pound force per square inch (psi)	6.984 757	$\times 10^{3}$	pascal (Pa)
Temperature			
degree Fahrenheit (^oF)	$[T(^oF) - 32]/1.8$		degree Celsius (^oC)
degree Fahrenheit (^oF)	$[T(^oF) + 459.67]/1.8$		kelvin (K)
Radiation			
curie (Ci) [activity of radionuclides]	3.7	$\times 10^{10}$	per second (s^{-1}) [becquerel (Bq)]
roentgen (R) [air exposure]	2.579 760	$\times 10^{-4}$	coulomb per kilogram (C kg^{-1})
rad [absorbed dose]	1	$\times 10^{-2}$	joule per kilogram (J kg^{-1}) [gray (Gy)]
rem [equivalent and effective dose]	1	$\times 10^{-2}$	joule per kilogram (J kg^{-1}) [sievert (Sv)]

*Specific details regarding the implementation of SI units may be viewed at http://www.bipm.org/en/si/.

†Multiply the U.S. customary unit by the factor to get the international unit. Divide the international unit by the factor to get the U.S. customary unit.

COVER PAGE: DTRA FINAL REPORT

Award Number HDTRA1-10-1-0050

Year: 2015
Federal Agency to Which Report is Submitted: DTRA
Federal Grant Number Assigned by Agency: HDTRA1-10-1-0050

Project Title: Mathematical Approaches to WMD Defense and Vulnerability Assessments on Dynamic Networks

PI Information: J. Cole Smith, Professor and Chair
Email: jcsmith@clemson.edu
Phone: (864) 656-4716

Draft Submission Date: 7/6/2015
DUNS: 019361885
EIN Number: 59-6002052

Recipient Organization: University of Florida, Engineering
Address: 339 Weil Hall; Gainesville, FL 32611

Grant Period (mm/dd/yyyy): **Start Date:** 4/12/2010 **End Date:** 8/11/2015
Reporting Period End Date (mm/dd/yyyy): 6/30/2015
Report Term or Frequency: Final

Final Report for HDTRA-10-1-0050
Mathematical Approaches to WMD Defense and Vulnerability Assessments on Dynamic Networks
PI: J. Cole Smith

I. Abstract

This research addresses the study of Network Adaptability from Weapons of Mass Destruction (WMD) Disruption and Cascading Failures. The general theme of the research studies the problems of combating disruptions due to WMD (C-WMD) on networks, taking into account real-world dynamics on critical infrastructures that are vulnerable to such attacks. Our research examines new modes of attack that are relevant to WMD analysis, and considers networks that are overlays of several individually functioning (but related) structures such as telecommunications, transportation, and electricity grids. More importantly, we analyze network dynamics that include cascading failures due to capacities on nodes and links, decentralized behavior due to independent agents on these networks, and adaptive recovery schemes in telecommunications settings. Thus, our analysis goes far beyond simple connectivity measures of graphs as a metric of damage, and considers much more relevant forms of disruption that could occur on our nation's infrastructure.

To combat these attacks, we examine the prevention and mitigation of attacks via intelligent mobile agent deployment, and via robust system design and recovery techniques that respond to worst-case WMD attacks. As a result, we demonstrate how to most effectively deploy existing and emerging technologies in concert with one another. Finally, we demonstrate how to design robustly connected networks that efficiently respond to massive localized outages. The proposing team utilizes techniques that hybridize mixed-integer programming, approximation theory, and game theory in order to guarantee desired levels of network operability.

II. Objective

The research seeks to develop theory and algorithms for problems that arise in combating disruptions due to Weapons of Mass Destruction (WMD) on networks. The science of countering WMD has shifted in the last decade, and now tends to focus on a system-wide integration of fortification resources, strategically deployed to combat an intelligent adversary. Despite the growing threat of WMD attacks, coinciding with a recent proliferation of network interdiction studies, optimal interdiction and defense studies against WMDs is still an area of vital research importance due to the number of open questions that exist in this realm. Our proposal identified several key goals to bridge the gap between current research network interdiction studies, and the theory and algorithms needed to tackle the countering of WMD attacks.

- Develop new interdiction models and algorithms in which an interdictor destroys a set of arcs that are geographically close to one another. In contrast to previous research, which assumes that individual disparate pieces of the network can be simultaneously attacked, our approach is directly applicable to WMD analysis and provides a far more realistic assessment of network vulnerabilities.

- Study dynamic networks, noting that the state of current and emerging networks increasingly adopts this paradigm. We also embed considerations of uncertain attack and defense effectiveness in our models for this goal to further improve the realistic nature of our models.

- Examine decentralized recovery techniques that many critical networks adopt under disruptions.

- Synthesize the results of the prior goals and consider fortification problems on networks under adaptive and distributive recovery schemes.

- Focus on optimally detecting and containing cascading failures, especially in the case of attacks designed to propagate across networks (including, e.g., biological agents).

- Apply our algorithms on test beds that simulate real-world telecommunication, transportation, and power grids.

- Develop compact formulations for "generalized network interdiction problems," in which clusters having a special structure can be deleted from a network, rather than just individual nodes or links.

- Identify new methods for critical network (node) element detection in interdependent systems, especially those having cascading failures.

- Explore dynamic interdiction problems, in which two-stage games are replaced by ongoing games in which an adversary destroys links, and the network operator can rebuild links over time.

- Examine the problem in which edges and clusters can be removed and are deemed as critical elements.

- Determine how to couple interdependent networks in order to create a robust system of networks. Given the presence of various networks, this goal seeks to determine which edges should be constructed to mitigate the deleterious effects of cascading failures.

III. Approach

The approach taken in this proposal uses a combination of interdiction algorithms, global optimization, combinatorial analysis, approximation algorithms, and decentralized computing. While the technical details are best left to the papers themselves, the general approach that we employ to address these problems is explained below.

The interdiction problems we consider involve two players: An attacker and a defender. The attacker moves first, and with knowledge of the attacker's move, the defender moves next. The defender is usually trying to optimize some objective over the network, such as finding a shortest origin-destination path, maximizing flow between two terminals, or maximizing network reliability. Therefore, the attacker seeks to inhibit the defender's objective, e.g., by maximizing his shortest path, or minimizing his maximum flow. The defender may in some situations act first in a fortification stage (producing so-called defender-attacker-defender models). Otherwise, we refer to the problem as attacker-defender models.

If the defender makes decisions in a centralized manner (by controlling all aspects of his actions, with full knowledge of the system, and with the ability to jointly utilize all of his resources), then these problems can be stated as "min-max" (or max-min) optimization problems: The attacker acts with the assumption that the defender will act optimally given the attacker's actions. (Moreover, in some versions of our problem, the defender may actually pre-empt certain attacker options with an initial fortification action.)

On the other hand, the defender may be *decentralized*, in the sense that network resources (e.g., monitors or communication hubs) are not jointly controlled by some unified force, but instead react only to local information that it can access. In this case, the defender is said to be adaptive. In this case, the defender is generally incapable of making an optimal response, but would instead have autonomous recovery protocols established throughout the network (e.g., as would be required in communication restoration).

One aim of this research seeks to piece together network complexity theory with nonlinear optimization theory and basic Euclidean geometry analysis, in order to tackle the class of network interdiction and optimization problems over continuous spaces. Furthermore, we have made very significant findings with respect to interdiction and competition arising in discrete games over networks. This includes the path planning problems mentioned above along with graph disconnection and clustering challenges.

Another aim is to develop a framework for designing decentralized algorithms that recover from network failures (perhaps, but not necessarily, due to intentional disruptions). Our methods are called *adaptive approximation algorithms*, because they do not assume the presence of a centralized optimizing entity (hence,

adaptive), but yet still perform with some guarantee of near-optimality despite their distributed nature. This class of methods is particularly vital in networks that are dynamic and evolving.

IV. Work Accomplished:

A summary of some of our primary work accomplishments is presented in Section IV.1, with a list of specific accomplishments given in Section IV.2.

IV.1 Overview

Geographical interdiction. One major research thrust regards network vulnerability assessments with geographical interdiction. In fact, our proposed geometric interdiction problems have proven to be extremely difficult to solve, and evidently require a refine-and-partition type of strategy to approximate. The tasks at hand are to provide tight lower- and upper-bounding strategies for these problems, solving a series of subproblems that are relatively tractable. In 2013, we reported that this paper was accepted for publication in *Networks*, and it appeared in 2014. Recently, this paper was awarded the Glover-Klingman prize for best paper appearing in a 2014 issue of *Networks*.

Also, regarding progress on geometric optimization problems, a study on extending the lifetime of wireless sensor networks spawned papers in *INFORMS Journal on Computing*, *IEEE Transactions on Mobile Computing*, and *IIE Transactions*. The latter journal is the flagship journal of the Institute of Industrial Engineers, and their professional magazine featured our work (Romich et al. 2015a) in the May 2015 issue of their *Industrial Engineer* magazine.

Understanding network structures can reveal behavioral functions as well as the vulnerability of the network. Knowledge of this structure provides us not only crucial information about network principles, but also key insights into designing more effective algorithms for practical problems. In particular, fathoming the interplay of different network elements as part of network evolution enabled by dynamics of the network leads to better understanding of the vulnerability of network elements. Many practical problems on complex networks, such as routing strategies in MANETs, sensor reprogramming in WSNs and worm containment in online social networks (OSNs) share an ubiquitous, yet interesting feature in their organizations: community structure. However, understanding this interesting feature is extremely challenging on dynamic networks where changes to their topologies are frequently introduced, and especially when network communities in reality usually overlap with each other.

We investigate network vulnerability based on communities obtained from underlying topology, which has not been investigated in the literature. To achieve

these goals, we introduce several new models based on the observation that most complex networks exhibit a network modular property. A network module may represent a functional group, a component, or an entity within the network system and the correlation among modules can model and describe the interplay between network components.

We have included the dynamics and evolution of a network in the analysis of inherent modules, which is a new task that has not yet been well investigated. The study of such evolution requires computing modules of the network at different time instances. However, identifying network modules in each state of the network from scratch may result in prohibitive computational costs, particularly in the case of highly dynamic networks. In addition, it may be infeasible in the case of limited topological data. In this regard, the study requires us to solve many interesting problems such as:

(a) How to devise new measures and methods for computing network modules that are robust to changes,
(b) How to easily update the modules once the changes occur without re-calculating them from scratch, and
(c) How sensitive the community structure is with respect to the failures of nodes and edges.

These studies aid in understanding the evolution of network components, and provide a robust solution for many applications which are sensitive to the structure of network communities.

Next, with respect to critical element selection in networks, our team has published in *IEEE Transactions on Reliability* on the topic of the robustness of power-law networks. Dr. Thai has also been active on this research topic, with work appearing in the *INFOCOM* and *GLOBECOM* conference proceedings.

We have also focused on the problem of removing cliques that maximally disconnect a network. This problem is called the Critical Clique Problem (CCP). We give special attention to variations of the CCP where different objectives functions are used, such as the number of pairwise connections, the size of the largest component, and the total number of residual components, among others. In addition, we are also interested in solving this problem when several conditions over the cliques are required, including upper and lower bounds on the size of the cliques. Finally, we are also adapting the proposed techniques for detecting other critical structures as well. See Walteros and Pardalos (2012) for more, and also Shen and Smith (2012a, 2012b) for related node-deletion studies. We also published several theoretical and algorithm results regarding these problems over the funding period, particularly in a pair of book chapters and in the journal *Theoretical Computer Science*.

We also examine Stackelberg problems (attacker-defender, or defender-attacker-defender) in which both players are faced with NP-hard problems. These problems

are notoriously difficult to solve and have resisted practical solution efforts thus far. Our goal is also to expand the scope of problems that can benefit from our research, especially in communication and sensor network settings. Another application area arises in civilian business scenarios that serve as proxies for military settings (e.g., models that can equivalently describe marketing challenges or military defense strategies).

Toward this goal, we have proposed a substantially new approach for problems that take place in defender-attacker-defender settings. The salient concepts can be presented in the attacker-defender max-min problem. One frequently solves this type of problem by *assuming that the follower's problem is convex*, taking the (strong) mathematical programming dual of the follower's problem, and combining the follower's problem (now converted to a maximization problem) with the leader's problem. If the resulting combined problem is nonlinear, which is typically the case, then some linearization strategy can be applied to convert the problem to a mixed-integer linear programming problem (MILP). This approach suffers from two drawbacks. One, the resulting MILP is typically very difficult to solve. Two, the italicized assumption above is limiting, and precludes the analysis of interdiction problems in which the follower solves, e.g., a discrete optimization problem. (One such study was accepted for publication in 2015. This paper, Tang et al. (2015), regards theoretical foundations for solving two-stage integer interdiction problems, and provides provably optimal algorithms for their solution. Those algorithms, however, are not scalable to large-scale instances.)

We have invented an entirely different concept behind solving these problems, which involves sampling potential defender solutions, and playing the game only over this sample. The sample space is then adaptively expanded until optimality can be proven. The algorithm we propose is not only capable of solving problems in which the follower's problem is an integer program, but our computational experiments show that it far more effective in solving traditional interdiction problems. For instance, consider the solution of shortest-path interdiction problems with fortification (SPIF): The defender first fortifies arcs against attack, the attacker then damages several undefended arcs (thus extending their interdiction costs), and the defender then selects a shortest path on the remaining network. We compared our algorithm to the current state-of-the-art algorithm for these problems. On a set of sixty 900-node instances, we obtained optimal solutions in 1.7 seconds on average, as opposed to 766.9 seconds on average using the current state-of-the-art, a speedup of 450 times. (The maximum computation times were 7.5 seconds for our approach, and 12,728.8 seconds for the incumbent approach.) These results are particularly exciting, because the SPIF is a very well-studied problem. The details are included in Lozano and Smith (2015).

Although it is early, we regard this as a **potentially landmark algorithm**, with widespread implications. One, this algorithm us to attempt problems that have been far too difficult to consider prior to now, such as interdicting traveling salesman problems. Furthermore, we regard this as the key missing piece that we needed in

order to solve the more difficult patrolling problems proposed in this paper. Two, this idea leads to new possibilities in investigating related problems that are also very difficult. These include bilevel problems, in which the follower seeks to optimize an objective that does not necessarily entail hurting the leader, but such that the follower's decision impacts the leader's objective. Our preliminary results show a 10-fold speedup compared with the best previous algorithm for this problem, when implemented on the same machine.

The investigators have also pursued multilevel work on integer programs using different means than the ones above. Prince and Smith (2015) consider network location problems with fortification and interdiction, in which the inner problems are integer programs but can be reformulated using an expanded variable set as convex optimization problems. Tadayon and Smith (2015) examine robust optimization problem in the context of scheduling problems, which have heretofore gone unexamined. Hemmati and Smith (2015) actually consider bilevel instead of interdiction problems, and use a technique related to lifting inequalities to obtain a finitely convergent optimal algorithm. Another pair of studies that we have authored on securing a border under asymmetric information have appeared in the prestigious journals *Mathematical Programming* and *Naval Research Logistics*; Dr. Kelly Sullivan was the key student on those papers (supported by DTRA) and has taken a faculty position as an Assistant Professor at the University of Arkansas.

Finally, Drs. Thai and Smith collaborated on an influence propagation model, which has recently been accepted for publication in *Computational Optimization and Applications*. In this model, we consider a two-stage defender-attacker game that takes place on a network, in which the attacker seeks to influence as many nodes as possible. The defender acts first in this game by protecting a subset of nodes that cannot be influenced by the attacker. With full knowledge of the defender's action, the attacker can then influence an initial subset of unprotected nodes. As above, this defender-attacker game is difficult to optimize because the attacker's problem itself is NP-hard (since a special case of that problem corresponds to the dominating set problem). Our research provides new mathematical and computational insights for solving these problems within reasonable computational limits.

Dynamic network problems can also be viewed in the light of dynamic interdiction games. Here, the defender makes a set of moves, followed by an attack, which is followed by another round of defense/attack moves until the game is complete. For instance, in the dynamic shortest path interdiction (DSPI) game studied in Sefair and Smith (2015), the attacker damages some arcs (possibly none) the defender moves on an arc, and the game continues by alternating those two such actions until the defender reaches the terminal node. Note that for this discussion, the attacker is assumed to have some limit b of arcs that it can damage.

These problems are very difficult, and evidently do not even belong to NP. They are related to robust optimization and interdiction as follows. Interdiction puts the attacker at a disadvantage, by having to expend all of its possible attacks before the

defender moves. Robust optimization gives the attacker an advantage by allowing the attacker to damage arcs after the defender has committed to a path. Dynamic interdiction is somewhere in between, where the attacker metes out its b attacks in response to the path taken so far by the defender, and where the defender adapts its path in response to the remaining attacker budget and set of attacked arcs.

Although optimal solutions to this problem are extremely difficult to characterize in general, it is possible to create various restrictions and relaxations for the attacker that permit bounds on the optimal objective to be obtained in a reasonable amount of computational time.

Interestingly, the DSPI is solvable in polynomial time when b is fixed. By contrast, the dynamic sequential assignment problem with interdiction (DAI) involves a defender, who assigns one project at a time in each period, one period at a time, until all projects are assigned. The attacker increases the assignment cost (depending on the project-period pair), and the game is played in a dynamic fashion as above. A similarity between DAI and DSPI is that when interdiction is not present, both defender problems are solvable in polynomial time. However, when the interdictor is limited to a single attack, DAI becomes strongly NP-hard (and in fact inapproximable), whereas DSPI is solvable in polynomial time. Sefair and Smith are producing a paper on this topic currently; Sefair's dissertation was successfully defended on these and other topics in June 2015.

The military applications that we are developing here also translate directly to many other areas that benefit from network models, such as marketing, social network analysis, supply chain operations, and so on. The quantitative analysis that we are developing should be an important complement to the principles established in mathematics, physics, and other branches of engineering, since those methods form the cornerstone of algorithms that extend well beyond our current study. Perhaps surprisingly, our work in complex networks and clique detection is important in computational neuroscience, for instance in the study of brain trauma disorders.

IV.2 Technical Details

In this section of our report, we give technical details for ten of our most impactful studies supported by DTRA. These ten studies are available in the archival literature, and we are pleased to offer data and code from these studies to our colleagues.

1 Geometric Interdiction

This discussion is taken from Sullivan and Smith (2014), with all proofs relegated to that paper. We consider the interdiction of a capacitated network that exists in Euclidean space. Nodes in this network exist at a point in space, and (directed) arcs connect node pairs in a straight line. An opponent wishes to maximize flow from a source node to a sink node across the network, while an interdictor seeks to minimize the opponent's maximum flow by choosing multiple locations to attack. In this problem, attacks are made at points in (Euclidean) space. Damage is inflicted on each arc by reducing its capacity as a function of the distance from the midpoint of the arc to each attack. We refer to this problem as the *Euclidean maximum flow network interdiction problem* (E-MFNIP). While distance can be computed according to any norm, we focus on the case in which distances are computed by L_1-norms, which will enable us to derive mixed-integer linear programming formulations. Also, we assume that each arc capacity is a function of the distance from the center of the arc to the closest attack location. We provide mathematical programming-based approaches for solving E-MFNIP.

The motivation for studying this type of problem is due to the prevalence of networks in complex systems and the vulnerability of those systems to attack. For instance, networks that represent real-world transportation, logistics, and power grid systems have well-defined geographical characteristics. These systems are subject to disruptions that may damage multiple components in the same geographical region. Most network interdiction research neglects any geographical characteristics, focusing instead on identifying network components that are most critical to sustaining functionality. In contrast, our methodology explicitly accounts for the simultaneous failure of network components correlated by the physical location of an attack. Furthermore, it is not necessary to restrict the scope of problems to those facing intentional attacks. Physical systems are vulnerable to accidental disruptions due to non-malicious events such as natural disasters as well, and must be robust enough to withstand worst-case disasters. Hence, although there is no entity that actively seeks to maximize damage in such a setting, it is still necessary to understand the worst-case vulnerability of the system.

Let $G(N, A)$ be a maximum flow network having node set $N = \{1, \ldots, n\}$ and directed arc set $A \subseteq N \times N$, where node 1 is the source and node n is the sink. Let c_{ij} denote the capacity of arc $(i, j) \in A$.

The maximum flow (denoted ν) from node 1 to node n can be obtained by solving the following

model, which represents the dual of a traditional maximum flow model:

$$\nu = \min \sum_{(i,j)\in A} c_{ij} v_{ij}, \tag{1a}$$

$$\text{s.t. } u_i - u_j + v_{ij} \geq 0, \quad \forall (i,j) \in A, \tag{1b}$$

$$u_1 = 0,\ u_n = 1, \tag{1c}$$

$$v_{ij} \geq 0, \quad \forall (i,j) \in A. \tag{1d}$$

Here, u_i denotes the dual variable associated with the flow balance constraints for node $i \in N$, and v_{ij} denotes the dual variable associated with the capacity constraint on arc $(i,j) \in A$. We define $\mathcal{U}$ to be the feasibility set that encompasses constraints (1b)–(1d).

In our problem, the interdictor chooses an attack Z from some feasible set $\mathcal{Z}$, and arc capacities are computed according to a function $\delta_{ij} : \mathcal{Z} \to \mathbb{R}$ for all $(i,j) \in A$.

After the leader selects an interdiction $Z \in \mathcal{Z}$, the follower maximizes flow on the resulting network in which c_{ij} is replaced by $\delta_{ij}(Z)$. The interdictor's problem is then to choose an interdiction that minimizes the follower's maximum flow, which can be stated as:

$$\nu^\star = \min \sum_{(i,j)\in A} \delta_{ij}(Z) v_{ij},\ \text{s.t. } (u,v) \in \mathcal{U},\ Z \in \mathcal{Z}. \tag{2}$$

When δ_{ij} is defined nontrivially (i.e., not constant) for some $(i,j) \in A$, this model is nonlinear in the objective. However, when $Z = \hat{Z}$ is fixed, there exists an optimal solution to the reduced version of Model (2) in which $v_{ij} \in \{0,1\}$ for all $(i,j) \in A$. Imposing these restrictions, we can linearize the product in the objective function by defining variables w_{ij} for each $(i,j) \in A$ to represent the product $\delta_{ij}(Z)v_{ij}$, and adding the inequalities

$$w_{ij} \geq \delta_{ij}(Z) - M_{ij}(1 - v_{ij}), \quad \forall (i,j) \in A, \text{ and} \tag{3a}$$

$$w_{ij} \geq 0, \quad \forall (i,j) \in A, \tag{3b}$$

where M_{ij} is an upper bound on $\delta_{ij}(Z)$ over all $Z \in \mathcal{Z}$. The resulting formulation is a MIP that is linearly constrained when δ_{ij} is a linear function:

$$\nu^\star = \min \sum_{(i,j)\in A} w_{ij}, \tag{4}$$

$$\text{s.t. Constraints (3a) and (3b)},\ (u,v) \in \mathcal{U},\ Z \in \mathcal{Z},\ v_{ij} \in \{0,1\},\ \forall (i,j) \in A.$$

Specification to E-MFNIP In previous research, $Z \in \mathcal{Z}$ typically consists of $|A|$ decisions, each representing the amount of interdiction imparted on an arc. E-MFNIP differs in its definition of $\mathcal{Z}$: We assume that G has physical structure, residing in q-dimensional Euclidean space, and that at most K interdictions are made at points on and around G. In what follows, we adopt the following notational conventions:

1. For any positive integer s, we use the notation $[s]$ to refer to the set $\{1, \ldots, s\}$.

2. We use Z to refer to an element of $\mathcal{Z}$. Thus Z suffices to represent all K interdiction locations.

3. We use $\mathbf{z}$, $\hat{\mathbf{z}}$, $\hat{\mathbf{z}}^k$, and any other adaptation of bold $\mathbf{z}$ to refer to the coordinates of a potential interdiction location in $\mathbb{R}^q$.

Hence, an element $Z \in \mathcal{Z}$ may be represented as $Z = [\mathbf{z}^1|\cdots|\mathbf{z}^K]$, where $\mathbf{z}^k \in \mathbb{R}^q,\ \forall k \in [K]$, is the location of the k-th interdiction. For simplicity, we assume that the constraint set $\mathcal{Z}$ is given as

$$\mathcal{Z} = \{Z \in \mathbb{R}^{q\times K} : \mathbf{L} \leq \mathbf{z}^k \leq \mathbf{U},\ k \in [K]\}, \tag{5}$$

where $\mathbf{L}$ and $\mathbf{U}$ are q-vectors. Each node $i \in N$ is located at point $\mathbf{z}(i) \in \mathbb{R}^q$, and each arc center $(i,j) \in A$ is located at the point $\mathbf{z}(i,j) = 0.5(\mathbf{z}(i) + \mathbf{z}(j))$. (All nodes $i \in N$ are assumed to satisfy $L \leq \mathbf{z}(i) \leq U$.) We assume that the damage inflicted by an attack on an arc is a function of the distance between the arc and the attack location. For simplicity and model tractability, we compute distance using the Manhattan norm, i.e., $\|\mathbf{z}\|_1 \equiv \sum_{k=1}^{q} |\mathbf{z}_k|$. We observe here that the resultsreadily extend to other convex norms with suitable modifications. However, using (e.g.) the Euclidean norm instead of the Manhattan norm results in a nonlinear (convex) model that is modestly more difficult to solve. Note also that the Manhattan norm can be used to approximate Euclidean distance, but at the risk of overestimating distance by a ratio of up to $(2-\sqrt{2})/\sqrt{2}$.

For $\mathbf{z} \in \mathbb{R}^q$, define $d_{ij}(\mathbf{z}) = \|\mathbf{z} - \mathbf{z}(i,j)\|_1$ as the distance between $\mathbf{z}$ and arc (i,j). In our model, the capacity of arc (i,j) is determined by the distance of the closest attack to $\mathbf{z}(i,j)$. That is, capacity functions are of the form

$$\delta_{ij}(Z) = f_{ij}\left[\min_{k\in[K]} d_{ij}(\mathbf{z}^k)\right], \tag{6}$$

where $f_{ij} : \mathbb{R}_+ \to \mathbb{R}_+$ is a function that maps distance to capacity. When $K = 1$, this assumption is intuitive. When $K > 1$, this assumption is possibly conservative (from the interdictor's perspective) because only one of the K interdictions influences the capacity of each arc. This assumption may be justifiable, though, in situations where each arc is composed of several structural components that determine the arc's capacity, and where each component fails due to varying thresholds of disruptions. Thus, less severe disruptions (given by attacks that are farther away than the closest attack) are only capable of affecting the same components that failed due to the closest attack, and hence the arc's capacity depends only on the closest attack location.

E-MFNIP is the problem that results when $\mathcal{Z}$ and δ are defined as in (5) and (6), respectively. We define $\nu^\star{}_E$ as the optimal objective value to E-MFNIP.

We now specialize (4) to model E-MFNIP as a MIP having a convex continuous relaxation under a certain class of f-functions. We assume that f_{ij} is given as the minimum of some T functions, i.e.,

$$f_{ij}(d) = \min_{t\in[T]} g_{ij}^t(d), \tag{7}$$

where $g_{ij}^t : \mathbb{R}^+ \to \mathbb{R}^+$ is convex and nondecreasing for each $t \in [T]$. As we will demonstrate, this model allows us to construct piecewise-convex capacity functions of distance.

Define $h_{ij}^t = g_{ij}^t \circ d_{ij}$ and note that h_{ij}^t is a convex function because g_{ij}^t and d_{ij} are convex and g_{ij}^t is nondecreasing. Moreover, observe that δ_{ij} can be represented as

$$\delta_{ij}(Z) = \min_{t\in[T]} g_{ij}^t \left[\min_{k\in[K]K} d_{ij}(\mathbf{z}^k) \right] \tag{8a}$$

$$= \min_{t\in[T],k\in[K]} g_{ij}^t \circ d_{ij}(\mathbf{z}^k) \tag{8b}$$

$$= \min_{t\in[T],k\in[K]} h_{ij}^t(\mathbf{z}^k), \tag{8c}$$

where (8b) follows because g_{ij}^t is nondecreasing.

Define λ_{ij}^{tk} to equal 1 if the minimum in (8b) is achieved by $t \in [T]$ and $k \in [K]$ and $\lambda_{ij}^{tk} = 0$ otherwise. Using the λ-variables, δ_{ij} can be expressed as

$$\delta_{ij}(Z) = \min \sum_{t\in[T],k\in[K]} \left[h_{ij}^t(\mathbf{z}^k)\right] \lambda_{ij}^{tk}, \tag{9a}$$

$$\text{s.t.} \sum_{t\in[T],k\in[K]} \lambda_{ij}^{tk} = 1 \text{ and } \lambda_{ij}^{tk} \in \{0,1\} \ \ \forall t \in [T],\ k \in [K]. \tag{9b}$$

This expression could be substituted for δ_{ij} in Model (4) to obtain a valid formulation. Furthermore, observe that decreasing $\delta_{ij}(Z)$ relaxes (3a), which is the only place $\delta_{ij}(Z)$ appears in Model (4). Thus, the minimization operator can be dropped from (9), because an optimal solution will always exist in which $\delta_{ij}(Z)$ takes its smallest value allowed by (9). The resulting mixed-integer nonlinear program (MINLP) minimizes $\sum_{(i,j)\in A} w_{ij}$ subject to $(u,v) \in \mathcal{U}$, $Z \in \mathcal{Z}$, $v \in \{0,1\}^{|A|}$, and (3b) as well as

$$w_{ij} \geq \sum_{t\in[T],k\in[K]} \left[h_{ij}^t(\mathbf{z}^k)\right] \lambda_{ij}^{tk} - M_{ij}(1 - v_{ij}), \quad \forall (i,j) \in A, \tag{10a}$$

$$\sum_{t\in[T],k\in[K]} \lambda_{ij}^{tk} = 1, \quad \forall (i,j) \in A, \text{ and} \tag{10b}$$

$$\lambda_{ij}^{tk} \in \{0,1\}, \quad \forall (i,j) \in A,\ t \in [T],\ k \in [K], \tag{10c}$$

where w_{ij} naturally takes the least possible value allowed by (10a).

To linearize (10a), let ϕ_{ij}^{tk} represent the product $\left[h_{ij}^t(\mathbf{z}^k)\right] \lambda_{ij}^{tk}$ and introduce inequalities similar to (3) to obtain the following model, valid for any $\bar{M}_{ij}^t$ such that $\bar{M}_{ij}^t \geq h_{ij}^t(\mathbf{z}),\ \forall \mathbf{L} \leq \mathbf{z} \leq \mathbf{U},\ (i,j) \in A,\ t \in [T]$.

$$\nu^\star{}_h = \min \sum_{(i,j)\in A} w_{ij} \tag{11a}$$

$$\text{s.t. } (u,v) \in \mathcal{U},$$

$$w_{ij} \geq \sum_{t\in[T],k\in[K]} \phi_{ij}^{tk} - M_{ij}(1 - v_{ij}), \quad \forall (i,j) \in A, \tag{11b}$$

$$w_{ij} \geq 0, \quad \forall (i,j) \in A, \tag{11c}$$

$$\sum_{t\in[T],k\in[K]} \lambda_{ij}^{tk} = 1, \quad \forall (i,j) \in A, \tag{11d}$$

$$\phi_{ij}^{tk} \geq h_{ij}^{t}(\mathbf{z}^k) - \bar{M}_{ij}^{t}(1-\lambda_{ij}^{tk}), \quad \forall (i,j) \in A,\ t \in [T],\ k \in [K], \tag{11e}$$

$$\phi_{ij}^{tk} \geq 0, \quad \forall (i,j) \in A,\ t \in [T],\ k \in [K], \tag{11f}$$

$$v_{ij} \in \{0,1\}, \quad \forall (i,j) \in A, \tag{11g}$$

$$\lambda_{ij}^{tk} \in \{0,1\}, \quad \forall (i,j) \in A,\ t \in [T],\ k \in [K], \tag{11h}$$

$$Z \in \mathcal{Z}.$$

As before, Model (11) is a MINLP, but this time its continuous relaxation is convex (because all h-functions are convex). These tractable continuous relaxations can be used to solve (11) via, e.g., branch-and-bound so long as all f-functions can be expressed as (7).

Solving over a General Capacity Function. In many applications, the f-functions cannot be represented in the form of (7). In the remainder of this discussion, we outline a procedure for using Model (11) to lower-bound the optimal E-MFNIP objective value by defining appropriate piecewise-linear g-functions.

We assume the true δ-functions take the form of (6), with f being nondecreasing and concave. These assumptions intuitively match what one would expect from the behavior of a realistic interdiction: Capacity increases as the interdiction moves farther away from an arc (nondecreasing), and a small change in interdiction location is more likely to have a pronounced impact on nearby arcs than on far away arcs (concave). If f is also piecewise-linear, Model (11) becomes an exact model using linear g-functions. If f is not piecewise-linear, g-functions can be selected to ensure that (11) provides a lower bound on the optimal E-MFNIP objective.

We now examine the following technique for approximating δ_{ij}-functions via piecewise-linear g_{ij}-functions. Suppose breakpoints $a_0, \ldots, a_T$ are given such that $0 = a_0 < a_1 < \cdots < a_T$, and $d_{ij}(\mathbf{z}) \leq a_T$ for all $\mathbf{L} \leq \mathbf{z} \leq \mathbf{U}$. For each $t \in [T]$, let g_{ij}^t be the linear function that approximates f_{ij} and is tight at a_{t-1} and a_t, i.e.,

$$g_{ij}^t(d) = \frac{f(a_t) - f(a_{t-1})}{a_t - a_{t-1}}(d - a_{t-1}) + f(a_{t-1}). \tag{12}$$

Theorem 1 *Let f be concave and nondecreasing, and suppose g_{ij}^t is given as in* (12)*. Then* (11) *defined with $h_{ij}^t = g_{ij}^t \circ d_{ij}$ provides a lower bound for E-MFNIP, i.e., $\nu^\star{}_h \leq \nu^\star{}_E$.*

Remark 1. It is possible to refine the g-functions dynamically within the solution of (11) to obtain a better approximation for E-MFNIP. These functions can be dynamically adjusted by adding breakpoints and segments to refine our approximation of the f-functions, based on solutions obtained from previous relaxations (e.g., as pioneered by Falk and Soland in 1969). However, this approach requires the iterative solution of the lower-bounding integer programs (11). Our experience indicates that each integer program (11) is itself very difficult to solve. The computational difficulties associated with this approach motivate an alternative algorithm below. □

Discrete Models. As an alternative, we also develop a methodology for generating solutions to E-MFNIP based on replacing the continuous set $\mathcal{Z} \subseteq \mathbb{R}^{q \times K}$ with a discrete set of locations, a subset of which will be attacked by the leader. The resulting integer program is a restriction of (4); therefore, all feasible solutions to this problem yield upper bounds on $\nu^\star{}_E$. We can then propose a modification of this model that provides a lower bound on $\nu^\star{}_E$ by relaxing the capacity functions.

Define $R = \{\mathbf{z} \in \mathbb{R}^q : \mathbf{L} \leq \mathbf{z} \leq \mathbf{U}\}$ and consider a model that selects K attacks from among a finite set of candidate locations $\{\bar{\mathbf{z}}^p\}_{p \in P}$ indexed over a set P, where $|P| \geq K$ and $\bar{\mathbf{z}}^p \in R$, $\forall p \in P$. (Henceforth, P will be used to index a set of candidate locations, and the $\bar{\mathbf{z}}$-notation will refer exclusively to the coordinates of a candidate location.)

Suppose interdictions are placed at locations $S \subseteq P$, where $|S| = K$. From (6), the resulting capacity of arc $(i,j) \in A$ is given by $f_{ij}[\min_{p \in S} d_{ij}(\bar{\mathbf{z}}^p)] = \min_{p \in S} f_{ij}[d_{ij}(\bar{\mathbf{z}}^p)]$, because $f_{ij} : \mathbb{R}_+ \to \mathbb{R}_+$ is nondecreasing. Given P, values $c^p_{ij} \equiv f_{ij}[d_{ij}(\bar{\mathbf{z}}^p)]$ can be computed a priori for each $p \in P$ and treated as constants; thus, the objective is to select $S \subseteq P$ that minimizes the maximum flow over a graph in which the capacity of arc (i,j) is $\min_{p \in S}\{c^p_{ij}\}$. We refer to this *discretized* problem as DE-MFNIP. To formulate DE-MFNIP as a MIP, let variables y^p equal one if location $p \in P$ is attacked and zero otherwise. The y-variables are constrained to be elements of the set $Y = \{y \in \{0,1\}^{|P|} : \sum_{p \in P} y^p = K\}$. For each $(i,j) \in A$, define $\bar{c}_{ij} \equiv \max_{p \in P}\{c^p_{ij}\}$ and observe that the capacity of arc (i,j) is $\min_{p \in P}\{\bar{c}_{ij} - (\bar{c}_{ij} - c^p_{ij})y^p\}$. Given $\hat{y} \in Y$, the follower solves its maximum flow problem with capacity constraints replaced by $x_{ij} \leq \bar{c}_{ij} - (\bar{c}_{ij} - c^p_{ij})\hat{y}^p$, $\forall p \in P$, where x_{ij} is a variable specifying how much flow traverses arc (i,j). Define $\nu_{DE}(y)$ as the maximum flow resulting from any $y \in Y$. The value of $\nu_{DE}(y)$ can be obtained alternatively by solving the following dual model, where γ^p_{ij} is the dual variable associated with the new capacity constraints:

$$\nu_{DE}(y) = \min \sum_{(i,j) \in A} \sum_{p \in P} [\bar{c}_{ij} - (\bar{c}_{ij} - c^p_{ij})y^p]\gamma^p_{ij}, \tag{13a}$$

$$\text{s.t. } (u,v) \in \mathcal{U}, \tag{13b}$$

$$\sum_{p \in P} \gamma^p_{ij} = v_{ij}, \ \forall (i,j) \in A. \tag{13c}$$

Model (13) is a linear program given y; thus, if (13) has an optimal solution, it must have one that is complementary slack with an optimal solution to its dual. Hence, adding the constraints $\gamma^p_{ij} \leq y^p$, $\forall (i,j) \in A$, $p \in P$, does not change the optimal objective value. (To see this formally, suppose that $\hat{y}^p = 0$ but $\hat{\gamma}^p_{ij} = 1$ in some optimal solution. Let $p' \in P$ be such that $\hat{y}^{p'} = 1$. By setting $\hat{\gamma}^p_{ij} = 0$ and $\hat{\gamma}^{p'}_{ij} = 1$ instead, we retain feasibility while reducing the objective value of the solution. Hence, it is valid to set $\gamma^p_{ij} \leq y^p$.) This relationship also permits us to set $\gamma^p_{ij} y^p = \gamma^p_{ij}$, because $\gamma^p_{ij} \leq y^p$ and $y^p \in \{0,1\}$. We now model DE-MFNIP as the following MIP:

$$\text{DE}(P)\text{: } \nu^\star{}_{DE}(P) = \min \sum_{(i,j) \in A} \sum_{p \in P} c^p_{ij}\gamma^p_{ij}, \tag{14a}$$

$$\text{s.t. } (u,v) \in \mathcal{U}, \tag{14b}$$

$$\sum_{p \in P} \gamma^p_{ij} = v_{ij}, \ \forall (i,j) \in A, \tag{14c}$$

$$0 \leq \gamma_{ij}^p \leq y^p, \ \forall (i,j) \in A, \ p \in P, \tag{14d}$$

$$y \in Y. \tag{14e}$$

Integer restrictions need not be placed on variables u, v, and γ in Model (14), as proven in the following theorem.

Theorem 2 *There exists an optimal solution to Model* (14) *such that all u-, v-, and γ-variables are binary-valued.*

Theorem 3 *If $\bar{\mathbf{z}}^p \in R, \ \forall p \in P$, then $\nu^\star{}_{DE}(P) \geq \nu^\star{}_E$.*

For $a \geq 0$, the function $c_{ij}^p(a) \equiv f_{ij}\left[\max\{0, d_{ij}(\bar{\mathbf{z}}^p) - a\}\right]$ provides a lower bound for $c_{ij}^p (= f_{ij} \circ d_{ij}(\bar{\mathbf{z}}^p))$ because f_{ij} is nondecreasing. By replacing $f_{ij}(d)$ in (14) with $f_{ij}\left[\max\{0, d-a\}\right]$, we obtain a relaxed model that will be useful in solving E-MFNIP. The function $f_{ij}\left[\max\{0, d-a\}\right]$ lower-bounds $f_{ij}(d)$ by "stretching" the true capacity function away from the attack location by radius a. Let radii $\alpha^p \geq 0$ be given for each $p \in P$ and consider the model DE(P, α) that results from replacing c_{ij}^p in Model (14) with $c_{ij}^p(\alpha^p)$:

$$\text{DE}(P,\alpha)\text{: } \nu^\star{}_{DE}(P,\alpha) = \min \sum_{(i,j)\in A} \sum_{p \in P} c_{ij}^p(\alpha^p)\gamma_{ij}^p, \tag{15}$$

$$\text{s.t. Constraints (14c)–(14e).}$$

When $\alpha^p = 0, \ \forall p \in P$, DE$(P, \alpha)$ is identical to Model (14); thus, $\nu^\star{}_{DE}(P) = \nu^\star{}_{DE}(P, 0)$.

DE-MFNIP is equivalent to the version of E-MFNIP that would result from restricting each $\mathbf{z}^k$ to come from $\{\bar{\mathbf{z}}^p\}_{p\in P}$ instead of from R. When $\alpha^p > 0$, we have that $c_{ij}^p(0) \geq c_{ij}^p(\alpha^p), \ \forall p \in P$ and $(i,j) \in A$. This raises the following question: How large must α^p be before we can guarantee that $\nu^\star{}_E \geq \nu^\star{}_{DE}(P, \alpha)$? To address this question, for $p \in P$ and $a \geq 0$, define $B(p, a)$ as the (one-norm) ball around $\bar{\mathbf{z}}^p$ with radius a, i.e., $B(p,a) = \{\mathbf{z} \in \mathbb{R}^q : \|\bar{\mathbf{z}}^p - \mathbf{z}\|_1 \leq a\}$. Also, given any finite set of candidate points $\hat{P}$ and a vector $\alpha \in \mathbb{R}_+^{|\hat{P}|}$, define $B(\hat{P}, \alpha) = \cup_{p\in\hat{P}} B(p, \alpha^p)$.

Theorem 4 *Suppose $\{\bar{\mathbf{z}}^p\}_{p\in P}$ and $\alpha \in \mathbb{R}_+^{|P|}$ are given such that $\bar{\mathbf{z}}^p \in R, \ \forall p \in P$. If $R \subseteq B(P, \alpha)$, then $\nu^\star{}_E \geq \nu^\star{}_{DE}(P, \alpha)$.*

Combining the results of Theorems 3 and 4, we have that $\nu^\star{}_{DE}(P, \alpha) \leq \nu^\star{}_E \leq \nu^\star{}_{DE}(P)$ for any α and P such that $\{\bar{\mathbf{z}}^p\}_{p\in P} \subset R \subseteq B(P, \alpha)$. Under modest assumptions on the f-functions, P and α may be chosen to guarantee that $\nu^\star{}_{DE}(P) - \nu^\star{}_{DE}(P, \alpha)$ is arbitrarily small (providing arbitrarily tight upper and lower bounds for $\nu^\star{}_E$), as proven in the following theorem.

Theorem 5 *Let $\{\bar{\mathbf{z}}^p\}_{p\in P}$ and $\alpha \in \mathbb{R}_+^{|P|}$ be given such that $\bar{\mathbf{z}}^p \in R, \ \forall p \in P$, and let $a > 0$ be given such that $\alpha^p \leq a, \ \forall p \in P$. For each $(i,j) \in A$, suppose f_{ij} is Lipschitz continuous and define $\tau_{ij} \geq 0$ as the corresponding Lipschitz constant. Also, define $\tau^\star$ as the weight of the maximum weighted 1–n cut over $G(N, A)$ with arc weights $\tau_{ij}, \ \forall (i,j) \in A$. Then $\nu^\star{}_{DE}(P) - \nu^\star{}_{DE}(P, \alpha) \leq a\tau^\star$.*

Theorem 5 guarantees that, with a large enough P-set and small enough α-values, the lower bound provided by ${\nu^\star}_{DE}(P, \alpha)$ is arbitrarily close to ${\nu^\star}_{E}$. However, obtaining a lower bound in this fashion is often impractical because the difficulty associated with solving DE-MFNIP grows quickly as P increases. We propose an alternative technique for developing tight lower bounds based on iteratively building the set P. Towards this end, we now prove a strengthened version of Theorem 5 that will be used to establish convergence of our algorithm.

Corollary 1 *Let P and α be given as in Theorem 5, and let $(\hat{u}, \hat{v}, \hat{y}, \hat{\gamma})$ be a binary-valued optimal solution to* DE(P, α). *The relationship ${\nu^\star}_{DE}(P) - {\nu^\star}_{DE}(P, \alpha) \leq a\tau^\star$ is valid for any a such that $a \geq \alpha^p$ for all $p \in P$ such that $\hat{y}^p = 1$.*

From Corollary 1, we know that the optimal objective value for DE(P, α) is within $a\tau^\star$ of ${\nu^\star}_{E}$, where a is the largest radius for any candidate point that was attacked in the optimal solution to DE(P, α). We now prove conditions under which modification of the set P guarantees that ${\nu^\star}_{DE}(P, \alpha)$ will not decrease. This result, when combined with Corollary 1, gives rise to an exact algorithm for solving E-MFNIP.

Theorem 6 *Let P^1, P^2, and $\{p^\star\}$ be disjoint index sets with $\bar{\mathbf{z}}^p \in R$ and $\alpha^p \geq 0$, $\forall p \in P^1 \cup P^2 \cup \{p^\star\}$, and suppose $R \subseteq B(P^1 \cup \{p^\star\}, \alpha)$. If $B(P^2, \alpha) = B(p^\star, \alpha^{p^\star})$, then ${\nu^\star}_{DE}(P^1 \cup \{p^\star\}, \alpha) \leq {\nu^\star}_{DE}(P^1 \cup P^2, \alpha) \leq {\nu^\star}_{E}$.*

An implication of Theorem 6 is that the contents of P can be modified in such a way that promotes an increase in ${\nu^\star}_{DE}(P, \alpha)$ without losing the property that ${\nu^\star}_{DE}(P, \alpha) \leq {\nu^\star}_{E}$. This result gives rise to a methodology for solving E-MFNIP, which we describe below.

Discretize-and-Refine Solution Methodology. We now describe how (14) can be used to solve instances of E-MFNIP. The method described here is applicable for general q, but our focus is on problems in which $q = 2$ (e.g., as would be the case in power grid, telecommunications network, and transportation settings).

First, specify an initial set of points $\{\bar{\mathbf{z}}^p\}_{p \in P_0}$ and values $\alpha_0^p \geq 0$, $\forall p \in P_0$, such that $R \subseteq B(P_0, \alpha_0)$. An initial lower bound for ${\nu^\star}_{E}$ can be obtained by solving DE(P_0, α_0). Solving this problem reveals an initial set of attack locations that are optimal for the relaxed problem. Next, set P_0 is modified (and renamed P_1) to consider a higher density of candidate locations surrounding locations that were optimal for DE(P_0, α_0). Corresponding α_1-values are assigned to the new elements of P_1 in accordance with the assumptions of Theorem 6 so that ${\nu^\star}_{DE}(P_0, \alpha_0) \leq {\nu^\star}_{DE}(P_1, \alpha_1)$. In this fashion, the lower bound is iteratively improved until it is within some acceptable tolerance gap of a known upper bound.

Upper bounds for ${\nu^\star}_{E}$ are obtained by solving DE(P) for any P (see Theorem 3). We now provide a formal description of our *discretize-and-refine* algorithm.

1. Select $\bar{\mathbf{z}}^p \in R$ and α_0^p for each $p \in P_0$, such that $R \subseteq B(P_0, \alpha_0)$. Set UB $= \infty$ and iteration counter $s = 0$. Let $\varepsilon > 0$ be a given tolerance parameter.

2. Solve DE(P_s) and obtain an optimal solution $(\bar{y}, \bar{u}, \bar{v}, \bar{\gamma})$ and upper bound $\nu^\star{}_{DE}(P_s)$. Define $\bar{P} = \{p \in P_s : \bar{y}^p = 1\}$.

3. If $\nu^\star{}_{DE}(P_s) <$ UB, then set UB $= \nu^\star{}_{DE}(P_s)$, and record an incumbent solution of $\{\bar{\mathbf{z}}^p\}_{p\in\bar{P}}$.

4. Solve DE(P_s, α_s) and obtain an optimal solution $(\hat{y}, \hat{u}, \hat{v}, \hat{\gamma})$ and lower bound $\nu^\star{}_{DE}(P_s, \alpha_s)$. Define $\hat{P} = \{p \in P_s : \hat{y}^p = 1\}$.

5. If UB $- \ \nu^\star{}_{DE}(P_s, \alpha_s) < \varepsilon$, then terminate the algorithm with near optimal attack locations given by the incumbent solution.

6. Set $P_{s+1} = P_s \setminus \hat{P}$. For each $\hat{p} \in P_s \setminus \hat{P}$, set $\alpha^{\hat{p}}_{s+1} = \alpha^{\hat{p}}_s$. For each $\hat{p} \in \hat{P}$:

   (a) Construct $2q$ new points defined over indices $\hat{p}^+_\ell$ and $\hat{p}^-_\ell$ for $\ell \in [q]$. For $\ell \in [q]$, define $\bar{\mathbf{z}}^{\hat{p}^+_\ell} = \bar{\mathbf{z}}^{\hat{p}} + (\alpha^{\hat{p}}_s/q)e_\ell$, and $\bar{\mathbf{z}}^{\hat{p}^-_\ell} = \bar{\mathbf{z}}^{\hat{p}} - (\alpha^{\hat{p}}_s/q)e_\ell$. (Note: e_ℓ is the q-vector with a one in component ℓ and zeros in all other components.) Add $\hat{p}^+_\ell$ and $\hat{p}^-_\ell$ to P_{s+1} for each $\ell \in [q]$.

   (b) Define $\alpha^{\hat{p}^+_\ell}_{s+1} = \alpha^{\hat{p}^-_\ell}_{s+1} = \alpha^{\hat{p}}_s(1 - 1/q), \ \forall \ell \in [q]$.

7. Set $s = s + 1$ and return to Step 2.

In moving from iteration s to iteration $s + 1$, the property that $R \subseteq B(P_{s+1}, \alpha_{s+1})$ is maintained because (i) $R \subseteq B(P_s, \alpha_s)$ and (ii) $B(\hat{p}, \alpha^s) \subseteq B(\{\hat{p}^+_\ell\}_{\ell\in[q]} \cup \{\hat{p}^-_\ell\}_{\ell\in[q]}, \alpha_{s+1}), \forall \hat{p} \in \hat{P}$. Thus, we guarantee that $\nu^\star{}_{DE}(P_s, \alpha_s) \leq \nu^\star{}_E, \ \forall s$, by Theorem 4. Moreover, $\cup_{\ell\in[q]}(B(\hat{p}^+_\ell, \alpha^{\hat{p}^+_\ell}_{s+1}) \cup B(\hat{p}^-_\ell, \alpha^{\hat{p}^-_\ell}_{s+1})) = B(\hat{p}, \alpha^{\hat{p}}_s)$; thus, $\nu^\star{}_{DE}(P_0, \alpha_0) \leq \nu^\star{}_{DE}(P_1, \alpha_1) \leq \cdots \leq \nu^\star{}_{DE}(P_s, \alpha_s) \leq \nu^\star{}_E, \ \forall s$, by Theorem 6. We now comment on the selection of P_0 and α_0, followed by a proof of convergence.

For $q = 2$ choosing P_0 as in Remark 2 guarantees $R \subseteq B(P_0, \alpha_0)$ with no overlap between the interior of $B(p, \alpha^p_0)$ and $B(\bar{p}, \alpha^{\bar{p}}_0)$, for any distinct $p, \bar{p} \in P_0$. (In fact, this property is preserved by the updates in Steps 6 and 7, implying for any iteration $s \geq 0$ that there is no overlap between $B(p, \alpha^p_s)$ and $B(\bar{p}, \alpha^{\bar{p}}_s)$, for any distinct $p, \bar{p} \in P_s$.) For $q > 2$, some overlap is inevitable. (For instance, in the case of $q = 3$, the of $B(p, a)$ is $4a^3/3$. However, p refines into six new points with radius $2a/3$: The combined volume of the resulting six L_1-norm balls is $6(4/3)(2a/3)^3 = 64a^3/27$, which is 78% greater than the volume occupied by $B(p, a)$.) This result induces some measure of inefficiency, but the algorithm remains convergent, as we now prove.

Theorem 7 *Suppose $r > 0$ is fixed and P_0 is chosen as described in Remark 2. If f_{ij} satisfies the assumptions of Theorem 5 for each $(i, j) \in A$ (and the definition of τ^* from Theorem 5 applies as well), then the discretize-and-refine algorithm converges to an ε-optimal solution within $|P_0| \sum_{k=0}^{B-1}(2q)^k$ iterations, where $B = \lceil \log_{1-1/q}(\varepsilon/\tau^\star r) \rceil$.*

We now illustrate our discretize-and-refine algorithm by applying it to a major fiber-optic network, obtained from Neumayer (2012). In the network, there are 170 nodes representing US cities and 230 undirected arcs connecting pairs of cities. The network is depicted in Figure 1.

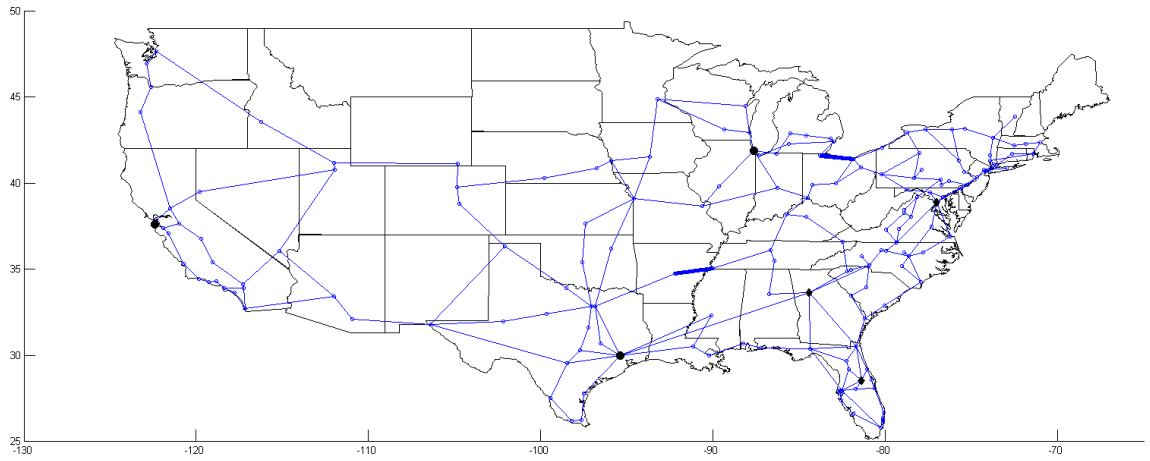


Figure 1: US Fiber Network.

Three nodes (Chicago, Houston, and San Francisco), indicated by large circles, are designated as sources and three nodes (Atlanta, Orlando, and Washington, D.C.), indicated by large diamonds, are designated as sinks. The network is converted into a directed network by replacing each undirected arc with two directed arcs, one in each direction. Latitude/longitude coordinates are used to establish node coordinates $\mathbf{z}(i)$, and arc capacities are computed using the functions $f_{ij}(d) = \min\{1, 0.2d\},\ (i,j) \in A$, where d corresponds to the distance to the nearest attack. Note that the nominal capacity of each arc is 1, and an arc's capacity can be decreased to zero if an interdiction is placed on its midpoint. In Figure 1, the thickened arcs (in Ohio and Arkansas) represent the arcs that would be interdicted under the classical MFNIP.

The node coordinates are all bounded within the coordinates $25°$W–$50°$W and $70°$N–$125°$N. We assume that the points within these bounds form a rectangle (which defines the feasible interdiction location set R) and generate P_0 and α_0 according to Remark 2 using $r = 5$. We then apply the discretize-and-refine approach to solve E-MFNIP when $K = 2$.

Figure 2 illustrates the initial candidate locations $\{\bar{\mathbf{z}}^p\}_{p \in P_0}$. In this figure, the L_1-norm ball with radius α_0^p has been drawn around each candidate solution, the center (indicated by a small "plus" symbol) of which marks the candidate solution itself. Using these candidate locations, we solve DE-MFNIP to compute lower and upper bounds for E-MFNIP. The solution to DE(P_0, α_0) yields a lower bound of 0.084 for E-MFNIP and includes interdictions at $(85°\text{W}, 40°\text{N})$ and $(90°\text{W}, 35°\text{N})$. This solution is depicted in Figure 2 by stars at the optimal interdiction locations. The lower-bounding solution is then used to refine the set of candidate solutions, producing the candidate set P_1 that will be used in the next iteration.

Figure 3 depicts P_1 along with the obtained lower-bounding solution, which places interdictions at $(87.5°\text{W}, 40°\text{N})$ and $(90°\text{W}, 32.5°\text{N})$. This solution is then refined to obtained P_2 (illustrated in Figure 4), which results in interdiction locations of $(85°\text{W}, 42.5°\text{N})$ and $(95°\text{W}, 30°\text{N})$ and improves the lower bound from 0.084 to 0.285. In each successive figure, the optimal attack locations from the previous iteration are replaced by four new candidate locations having smaller radii.

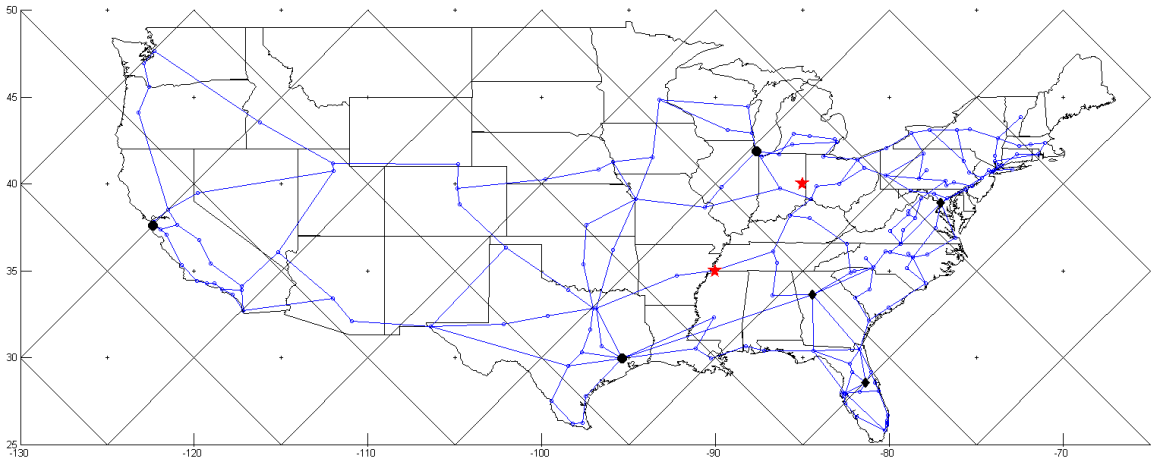


Figure 2: Initial candidate interdiction locations (LB $= 0.084$, UB $= 2.814$).

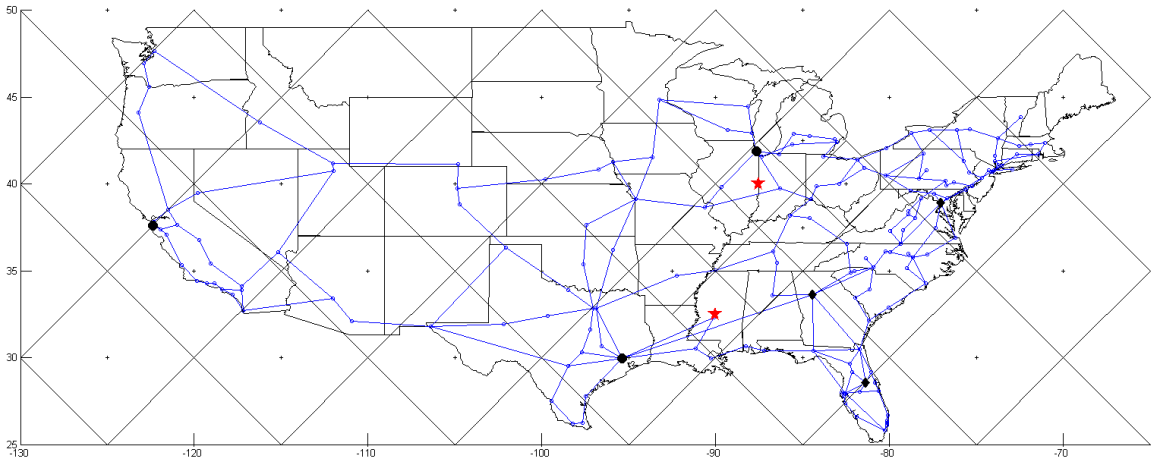


Figure 3: Candidate interdiction locations after one iteration (LB $= 0.285$, UB $= 2.38$).

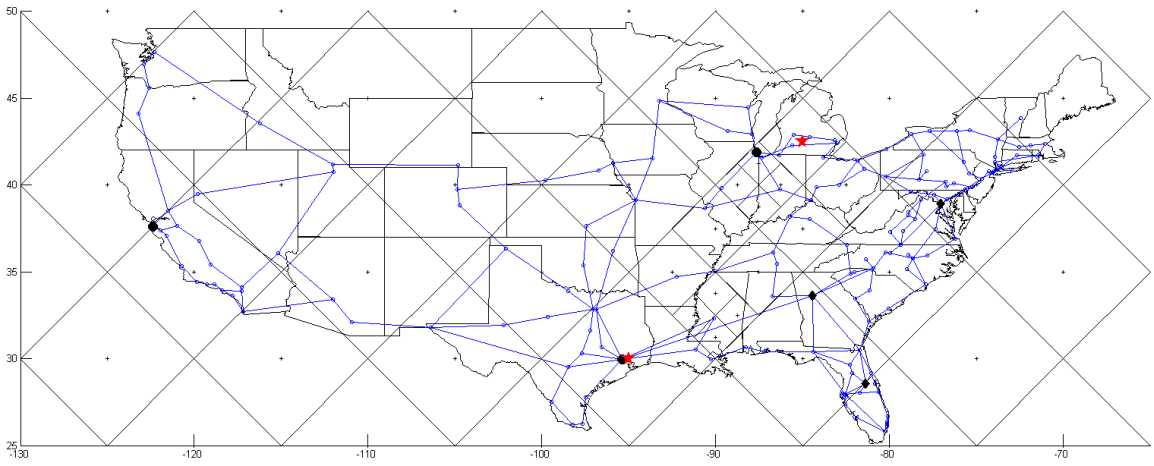


Figure 4: Candidate interdiction locations after two iterations (LB $= 0.575$, UB $= 2.356$).

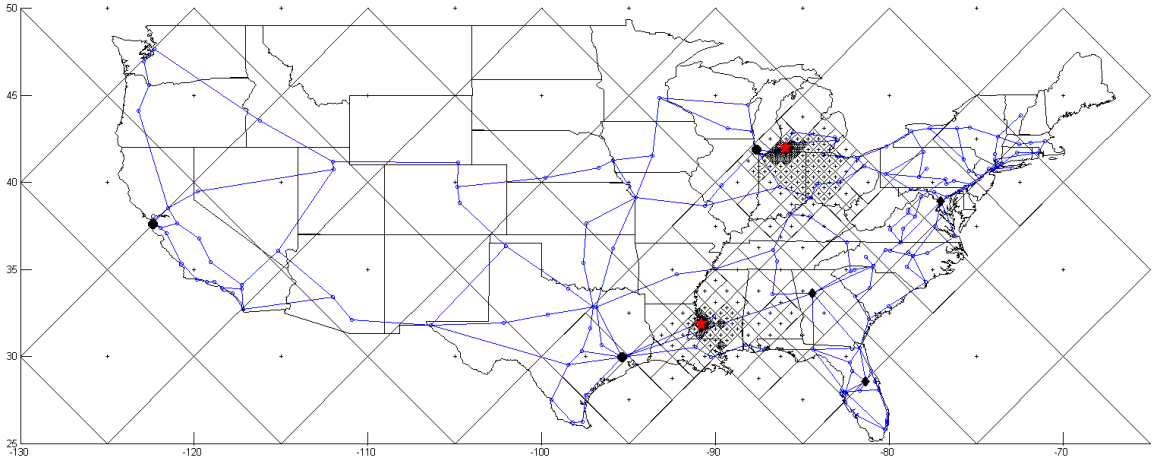


Figure 5: Near-optimal interdiction locations after 62 iterations (LB $= 1.941$, UB $= 1.957$).

This process is repeated until the upper and lower bounds are sufficiently close (within 1%, requiring 62 iterations). The resulting solution, illustrated in Figure 5, contains interdictions at $(85.9375°\text{W}, 41.9922°\text{N})$ and $(90.7812°\text{W}, 31.8359°\text{N})$, and produces a lower bound of 1.941. This example also illustrates the potential benefit of accounting for network geography in interdiction studies. Attacking the midpoints of the optimal MFNIP arcs (the thick arcs in Figure 1) results in an E-MFNIP objective of 2.785, which is 42% greater than the best known E-MFNIP solution.

2 Identifying Community Structures

We consider a network represented as an undirected graph $G = (V, E)$ consisting of $n = |V|$ vertices and $m = |E|$ edges. The *adjacency matrix* of G is denoted by $\boldsymbol{A} = (A_{ij})$, where A_{ij} is the weight of edge (i, j) and $A_{ij} = 0$ if $(i, j) \notin E$. We also denote the (weighted) degree of vertex i, the total weights of edges incident at i, by $\mathsf{deg}(i)$ or, in short, d_i.

Community structure (CS) is a division of the vertices in V into a collection of disjoint subsets of vertices $\mathcal{C} = \{C_1, C_2, \ldots, C_l\}$ where $\bigcup_{i=1}^{l} C_i = V$ and l is the total number of identified subsets. Each subset $C_i \subseteq V$ is called a *community* and we wish to have more edges connecting vertices in the same communities than edges that connect vertices in different communities. The *modularity* of $\mathcal{C}$ is the fraction of the edges that fall within the given communities minus the expected number of such fraction if edges were distributed at random. The randomization of the edges is done so as to preserve the degree of each vertex. If vertices i and j have degrees d_i and d_j, then the expected number of edges between i and j is $\frac{d_i d_j}{2M}$. Thus, the modularity, denoted by Q, is then

$$Q(\mathcal{C}) = \frac{1}{2M} \sum_{ij} (A_{ij} - \frac{d_i d_j}{2M}) \delta_{ij} \tag{16}$$

where M is the total edge weights and the element δ_{ij} of the membership matrix $\boldsymbol{\delta}$ is defined as

$$\delta_{ij} = \begin{cases} 1, & \text{if } i \text{ and } j \text{ are in the same community} \\ 0, & \text{otherwise} \end{cases}$$

The modularity values can be either positive or negative and the higher (positive) modularity values indicate stronger community structures. Therefore, the *maximizing modularity problem* asks us to find a division $\mathcal{C}$ which maximizes the modularity value $Q(\mathcal{C})$.

This problem is different from the partition problem as we do not know the total number of partitions l beforehand. Somewhat surprisingly, modularity maximization is still NP-complete on trees, one of the simplest graph classes.

Theorem 8 *Modularity maximization on trees is NP-complete.*

The proof has been presented in Dinh and Thai (2015), reducing from the Subset-Sum problem.

Exact Solutions. Although the problem is in NP class, efficient algorithms to obtain optimal solutions for small size networks are still of interest. We have presented an exact algorithm with a run time of $O(n^5)$ to the problem on uniform-weighted trees Dinh and Thai (2015). The algorithm is based on the dynamic programming, which exploits the relationship between maximizing modularity and minimizing the sum-of-squares of component volumes, where volume of a component S is defined as $\mathsf{vol}(S) = \sum_{v \in S} d_v$.

When the input graph is not a tree, we provided an exact solution based on the following Integer Linear Programming (ILP).

$$\max \quad \frac{1}{2M}\sum_{i,j} B_{ij}(i - x_{ij}) \tag{17}$$

$$\text{s.t.} \quad x_{ij} + x_{jk} + x_{ik} \geq 0 \quad \forall i < j < k \tag{18}$$

$$x_{ij} + x_{jk} + x_{ik} \geq 0 \quad \forall i < j < k \tag{19}$$

$$-x_{ij} + x_{jk} + x_{ik} \geq 0 \quad \forall i < j < k \tag{20}$$

$$x_{ij} \in [0, 1] \qquad i, j \in [1..n] \tag{21}$$

where $B_{ij} = A_{ij} - \frac{d_i d_j}{2M}$. Constraints (18), (19), and (20) are well-known triangle inequalities that guarantee the values of x_{ij} are consistent to each other. They imply the following transitivity: if i and j are in the same community and j and k are in the same community, then so are i and k. By definition, $x_{ii} = 0\ \forall i$ and can be removed from the ILP for simplification.

Therefore, the ILP has $3\binom{n}{3} = \theta(n^3)$ constraints, which is about half a million constraints for a network of 100 vertices. As a consequence, the sizes of solved instances were limited to few hundred nodes. Along this direction, we have presented a sparse metric, which reduces the number of constraints to $O(n^2)$ in sparse networks where $m = O(n)$.

Approximation Algorithms. When G is a tree, the problem can be solved by a polynomial time approximation scheme (PTAS) with a run time of $O(n^{\epsilon+1})$ for $\epsilon > 0$. The PTAS is solely based on the following observation. Removing $k - 1$ edges in G will yields k connected communities and $Q_k \geq (1 - \frac{1}{k})Q_{opt}$ where Q_k is the maximum modularity of a community structure with k communities, and Q_{opt} is the optimal solution. The PTAS algorithm for maximizing the modularity follows next.

Algorithm. PTAS for Maximizing modularity on Trees

1. Given $\epsilon > 0$, set $k = \lceil \frac{1}{\epsilon} \rceil$
2. $Q_k = 0, \mathcal{C}_k = \{V\}$
3. **for each** $X \subset E$ and $|X| < k$ **do**
4. Find connected component $C_1, C_2, \ldots, C_k$ in $T' = (V, E \backslash X)$
5. Let CS $\mathcal{J} = \{C_1, C_2, \ldots, C_k\}$
6. **if** $Q(\mathcal{J}) > Q_k$ **then**
7. $Q_k = Q(\mathcal{J})$
8. $C_k = \mathcal{J}$
9. Return C_k

When G having the degree distribution that follows the power-law, i.e., the fraction of nodes in the network having k degrees is proportional to $k^{-\gamma}$, where $1 < \gamma \leq 4$, the problem can be approximated to a constant factor for $\gamma > 2$ and up to an $O(1/\log n)$ when $1 < \gamma \leq 2$ (Dinh and Thai, 2013). We propose the approximation algorithm, Low-Degree Following (LDF) in this regard.

LDF decides for each vertex u, which neighbor to follow, and if u follows a neighbor v, the algorithm eventually assigns u and v to the same community. The algorithm follows three rules to assign each vertex one of the three labels *leader*, *member*, or *orbiter* as follows:

1. All members and orbiters have degree at most d_0, for some predefined parameter d_0.
2. There are only two types of following: a member follows a leader and an orbiter follows a member. This implies that members cannot follow each other and orbiters cannot directly follow leaders.
3. All neighbors of an orbiter must be members.

The algorithm uses three sets L, M, and O to store leaders, members, and orbiters, respectively, and an array p_i to store which neighbor vertex i follows. A vertex that does not belong to any of the sets L, M, or O is said to be unlabeled. Initially, L, M, and O are empty i.e. all vertices are unlabeled and $p_i = 0$ for all i.

At each step, the algorithm considers an unlabeled vertex i of degree at most d_0. If there is a neighbor $j \in N(i) \backslash M$ of i that is a leader or an unlabeled vertex, we add i to M and j to L, if necessary, and set i to follow j, $p_i = j$. Otherwise, all neighbors of i must be labeled with member, thus, we can add i to the set of orbiters O and set i to follow an arbitrary neighbor t.

Finally, two types of communities are formed. First, all the *members* that follow the same leader and all the *orbiters* that follow those *members* are assigned into the same community. Second, each unlabeled vertex forms a singleton community of size one. The union of all the communities is returned as the community structure $\mathcal{L}$.

The selection of d_0 is important to derive the approximation factor as d_0 needs to be a sufficient large constant that is still relative small to n when n tends to infinity. In an actual implementation of the algorithm, we have designed an automatic selection of d_0 to maximize Q. LDF can be extended to solve the problem in directed graphs (Dinh and Thai, 2013).

The details of this LDF algorithm is presented below.

Algorithm. Low-degree Following Algorithm (Parameter $d_0 \in \mathbb{N}^+$)

1. $L \leftarrow \emptyset, M \leftarrow \emptyset, O \leftarrow \emptyset, p_i = 0 \; \forall i = 1..n$
2. **for each** vertex $i \in V$ **do**
3. **if** $(k_i \leq d_0)$ & $(i \notin L \cup M)$ **then**
4. **if** $N(i) \setminus M \neq \emptyset$ **then**
5. Select a vertex $j \in N(i) \setminus M$
6. Let $M = M \cup \{i\}, L = L \cup \{j\}, p_i = j$
7. **else**
8. Select a vertex $t \in N(i)$
9. $O = O \cup \{i\}, p_i = t$
10. $\mathcal{L} = \emptyset$
11. **for each** vertex $i \in V \setminus (M \cup O)$ **do**
12. $C_i = \{i\} \cup \{j \in M \mid p_j = i\} \cup \{t \in O \mid p_{p_t} = i\}$
13. $\mathcal{L} = \mathcal{L} \cup \{C_i\}$
14. Return $\mathcal{L}$

Furthermore, in some cases, communities are sharing some nodes between them, referred as overlapping communities. That is, a person or a node can belong to more than one community. Therefore, we further designed an algorithm, namely DOCA (Detecting Overlapping Community Algorithm), to find overlapping network modules which required only one parameter, indicating the level of overlapping (Nguyen et al. 2011) denoted by the ration β.

DOCA consists of three different phases:

1. Detecting Local Communities
2. Combining Overlapping Communities
3. Revisit Unassigned Nodes

In the first phase, we identify the local communities by maximizing the internal density of each communities (Nguyen et al. 2011). The density based function can be defined as $\Psi(C) = \frac{|C^{in}|}{\binom{|C|}{2}}$ to identify a set C of nodes as a community. The more C approaches a clique of its size, the higher its density value $\Psi(C)$. C^{in} and C^{out} denote the set of edges having both endpoints in C and one endpoint in C, respectively.

The threshold on the internal density that suffices for C to be a local community is given by

$$\tau(C) = \frac{\sigma(C)}{\binom{|C|}{2}} \text{ where } \sigma(C) = \binom{|C|}{2}^{1-\frac{1}{\binom{|C|}{2}}} \tag{22}$$

We term a group of nodes $C \subseteq V$ a local community if its internal density exceeds a threshold determined based only on C's size, regardless to its external connections. Of course, a clique represents a perfect local community; however, we do not restrict our starting communities to be only cliques. Instead, we relax them to be "quasi-cliques" which can overlap with each other and whose detection can be done in an automatic fashion. The phase is described in Alg. 1.

Algorithm 1: Detecting Local Communities

Input : Network $G = (V, E)$
Output: Local (or raw) community structure $\mathcal{C} = \{C_1, C_2, \ldots, C_k\}$

```
C = ∅;
for (u, v) ∈ E and Com(u) ∩ Com(v) = ∅ do
    Let C_uv = N(u) ∩ N(v) ∪ {u, v};
    if Δ(C_uv) ≥ τ(C_uv) then
        C ← C ∪ C_uv;
        Update Com(u) and Com(v);
    end
end
```

As soon as the first procedure finishes, the raw network community structure can be pictured as a collection of (possibly overlapped) dense parts of the network together with outliers. As some of those dense parts can possibly share significant common substructures, we need to merge them if they are indeed highly overlapped. In order to do so, we introduce the overlapping score of two communities: $OS(C_i, C_j) = \frac{|I_{ij}|}{min\{|C_i|,|C_j|\}} + \frac{|I_{ij}^{in}|}{min\{|C_i^{in}|,|C_j^{in}|\}}$ where $I_{ij} = C_i \cap C_j$. Basically, $OS(C_i, C_j)$ values the importance of the common nodes and connections shared between C_i and C_j to the smaller community.

The second phase starts out by examining raw communities identified from the first phase in a bottom up manner. In this procedure, two communities C_i and C_j are combined if $OS(C_i, C_j) \geq \beta$. The algorithm is described in Alg. 2.

Algorithm 2: Combining Overlapping Communities

Input : Local network community structure $\mathcal{C}' = \{C'_1, C'_2, \ldots, C'_p\}$
Output: Combined community structure $\mathcal{C} = \{C_1, C_2, \ldots, C_t\}$

$C = \mathcal{C}'$, $Done \leftarrow$ False;
while *(! Done)* **do**
  $Done \leftarrow$ True;
  for $C_i \in \mathcal{C}$ **do**
    Let $N(C_i) = \{C_j \in \mathcal{C} | C_j \cap C_i \neq \emptyset\}$;
    for $C_j \in N(C_i)$ **do**
      if $OC(C_j, C_i) \geq \beta$ **then**
        $C_{min\{i,j\}} \leftarrow C_i \cup C_j$;
        $\mathcal{C} \leftarrow \mathcal{C} \backslash C_{max\{i,j\}}$;
        Update $Com(u)$ for all $u \in C_i$;
        $Done \leftarrow$ False;
      end
    end
  end
end

Even when the above two procedures are executed, there would still exist leftover nodes or edges due to their less attraction to the rest of the network. Because of its size constraint, the first procedure skips over tiny communities of sizes less than four and thus, may leave out some nodes unlabeled. These nodes will not be touched in the second phase since they do not belong to any local communities and consequently, will remain unassigned afterwards. Therefore, we need to revisit those nodes to either group them into appropriate communities or classify them as outliers based on their connectivity structures. Alternatively, this process can be thought of as a community trying to hire adjacent unassigned nodes which are similar to the host community. To this end, we need a community fitness function in order to quantify the similarity between a node u and a neighbor community C. We find the fitness function $F_S = \frac{|S^{in}|}{2|S^{in}|+|S^{out}|}$ for $S \subseteq V$. Taking into account this fitness function, a community C will keep hiring any unassigned adjacent vertex of maximum similarity in a greedy manner, provided the newly joined vertex does not shrink down the community's current fitness value. If there is no such node, C is defined as a final network community shown in Alg. 3. Simulations showed that this is the best one in the literature (Nguyen et al. 2011).

3 Adaptively Updating Dynamic Community Structures

Many real world networks are dynamic in nature where elements in the network (vertices and edges) evolve over time. In this project, We continued studying the adaptive identification of community structures, focusing on the following question: How to update the evolving community structures without re-computing it from scratch. In this approach, the community structure (CS) at

Algorithm 3: Revisit Unassigned Nodes

Input : The combined community structure $\mathcal{C}' = \{C'_1, C'_2, \ldots, C'_t\}$
Output: The final community structure $\mathcal{C} = \{C_1, C_2, \ldots, C_k\}$

```
C = C';
for u ∈ V and Com(u) = ∅ do
    Let NC(u) = {C_j ∈ C | u is adjacent to C_j};
    for C_j ∈ NC(u) do
        if F_{C_j ∪ {u}} ≥ F_{C_j} then
            C_j ← C_j ∪ {u};
            Com(u) ← Com(u) ∪ {j};
        end
    end
    if Com(u) = ∅ then
        Classify u as an outlier;
    end
end
```

time t is detected based on the community structure at time $t-1$ and the changes in the network, instead of recomputing it directly at time t without taking advantages of a current solution at time $t-1$. Along this direction, we have devised an adaptive approximation algorithm for this problem, published in Dinh et al. (2013). Indeed, the above LDF algorithm described in before can be enhanced to cope with this situation. At first LDF is run to find the base CS at time 0. Then at each time step, we adaptively follow and unfollow the nodes that violate the condition 3 in the LDF algorithm. We provide greater detail of this proposed algorithm, namely A^{3}CS, below.

A^{3}CS is a meta-algorithm that first calls A-Base algorithm to find the community structure $C^{(0)}$ of the first network snapshot $G^{(0)}$, then iteratively finds community structure $C^{(t)}$ at time point t by invoking the A-Adaptive algorithm. The two algorithms A-Base and A-Adaptive construct the community structure via assigning values for two arrays $label[i]$ and $follow[i]$.

Algorithm. A^{3}CS - Adaptive Approx. Alg. for CS

1. $C^{(0)} = \text{A-Base}(G^{(0)})$
2. **for** $t = 1$ to s **do**
3. $\quad C^{(t)} = \text{A-Adaptive}(C^{(t-1)}, \Delta G^{(t)})$

The meaning of $label$ and $array$ is as follow. Each node i is labeled with either $leader$, $follower$, or unlabeled (also denoted with $\varnothing$). For a node i labeled with $follower$, $follow[i]$ is the name of the leader that i follows. Precisely, we have

$$follow[i] = \begin{cases} i & \text{if } label[i] = leader \\ j \neq i & \text{if } label[i] = follower \ \& \ i \text{ follows } j \\ \varnothing & \text{if } label[i] = \varnothing \end{cases}$$

At any time point t, the community structure is given by the union of two types of communities: 1) all *followers* that follow the same *leader* are assigned into the same community; and 2) each unlabeled node forms a singleton community of size one.

At the heart of the proposed algorithms, the assigned labels satisfy the important properties stated in the following lemma.

Lemma 1 *At the end of the algorithms A-Base and A-Adaptive, the following properties hold.*

1. *All* low-degree *nodes i.e., nodes with degree at most d_0 for some predefined constant $d_0 > 0$, are labeled either with $leader$ or $follower$.*
2. *All* followers *are* low-degree *nodes.*
3. *Each* leader *is followed by at least one* follower*; and each* follower *follows exactly one* leader. *Thus* followers *will not follow each other or unlabeled nodes.*

The intuition to this lemma will be explained through the presentation of A-Base and A-Adaptive.

Algorithm. A-Base

1. $label[i] = \varnothing, follow[i] = \varnothing \; \forall i = 1..n$
2. Sorted nodes in non-decreasing order of degree.
3. **for each** vertex i with $k_i \leq d_0$ **do**
4. **if** $label[i] = \varnothing$ **then**
5. FOLLOW_NEIGHBOR(i)
6. Return $C^{(0)} = \langle follow \rangle$

A-Base. This algorithm finds the community structure of $G^{(0)}$ via labeling nodes in the network. Nodes are first sorted in a non-decreasing order of degree, and then, each *low-degree* and unlabeled node i selects one of its neighbors to follow using the FOLLOW_NEIGHBOR algorithm, in which the $label[i]$ and the $follow[i]$ are assigned accordingly. We can verify that all the properties in Lemma 1 hold at the end of A-Base.

Algorithm. A-Adaptive ($C^{(t-1)}, \Delta G^{(t)}$)

1. **for each** edge $(u, v) \in \Delta E^{(t)}$ **do**
2. Update degree of nodes u and v
3. **for each** vertex i appears in $\Delta G^{(t)}$ **do**
4. **if** $(k_i \leq d_0)$ & $(label[i] = \varnothing)$ **then**
5. FOLLOW_NEIGHBOR(i)
6. **else if** $(k_i > d_0)$ & $(label[i] = follower)$ **then**
7. UNFOLLOW(i)
8. Return $C^{(t)} = \langle follow \rangle$

A-Adaptive. This algorithm finds the community structure at time point t based on $C^{(t-1)}$ and $\Delta G^{(t)}$ - the previous community structure and the changes in the network. After updating the node degrees (lines 1 to 2), the algorithm checks all nodes that appear in $\Delta G^{(t)}$ and corrects all possible "mis-labeling" caused by the degree changes. Two "mis-labeling" cases are 1) low-degree and unlabeled nodes as the results of removing edges (or adding new nodes), and 2) follower nodes with the degrees higher than d_0 as the results of adding new edges/nodes. The two "mis-labeling" cases are corrected using FOLLOW_NEIGHBOR and UNFOLLOW algorithms, as shown in lines 4 to 7.

Algorithm. FOLLOW_NEIGHBOR(i)

1. $label[i] = follower$
2. **if** $\exists j \in N(i) : label[j] \neq follower$ **then**
3. **if** $label[j] = \varnothing$ **then** $label[j] = leader$
4. $follow[i] = j$
5. **else**
6. Select a random $j \in N(i)$
7. UNFOLLOW(j)
8. $follow[i] = j, label[j] = leader$
9. Update the modularity value.

FOLLOW_NEIGHBOR. This is the fundamental procedure in A^{3}CS. Given a node i, the algorithm identifies a neighbor j so that i can follow j without violating the properties of Lemma 1. Lines 3 and 4 explore the case when we can find a non-follower neighbor j of i. When all neighbors of i are followers, we first use the UNFOLLOW algorithm to make a neighbor j of i unlabeled or labeled it with $leader$, and only then we can let i follow j (lines 6 to 8).

Algorithm. UNFOLLOW(i)

1. Let $j = follow[i], label[i] = \varnothing$
2. **if** j has no followers **then**
3. **if** $k_j \leq d_0$ **then**
4. $follow[j] = i, label[j] = follower$
5. $label[i] = leader$
6. **else** $label[j] = \varnothing$
7. Update the modularity value.

UNFOLLOW. As briefly mentioned, the algorithm UNFOLLOW is invoked when we need to stop a node i from following its current leader j. This can usually be done by simply unlabeling i. The interesting case happens when i is the only follower of j and unlabeling i will make j a leader without followers (opposing the third property in Lemma 1). We handle this case by either unlabeling j or swapping i and j's labels together with making j follow i (lines 3 to 6).

Many real world networks in reality are highly dynamic and thus, their communities are not always disjoint from each other. Indeed, their communities often overlap with each other since some active nodes can participate in multiple groups at the same time, thereby reassemble the concept of overlapping community structure. Furthermore, most practical models for network problems evolve frequently over time due to the high dynamics of participating nodes. Although any slight change does not seem to have a significant effect on the network structure, the evolution of the complex network over a long duration might lead to an unpredictable transformation of its communities, particularly when they can overlap. In this context, we further investigated the evolution in overlapping community structure and provided the first adaptive algorithm to adaptively update the overlapping network modules. This work is published in Nguyen et al. (PLoS ONE and MOBICOM).

In order to reflect these changes to a complex network, its underlying graph model is frequently updated by either inserting or removing a node or a set of nodes, or an edge or a set of edges. Scrutiny into these events reveals that the introduction or removal of a set of nodes (or edges) can furthermore be decomposed as a collection of node (or edge) insertions (or removals), in which only a node (or only an edge) is inserted (or removed) at a time. Therefore, changes to the network at each time step can be viewed as a collection of simpler events whose details are as follow:

- *newNode* $(V + u)$: A new node u and its adjacent edge(s) are introduced
- *removeNode* $(V - u)$: A node u and its adjacent edge(s) are removed from the network.
- *newEdge* $(E + e)$: A new edge e connecting two existing nodes is introduced.
- *removeEdge* $(E - e)$: An edge e in the network is removed.

Our adaptive framework, namely Adaptive Finding Overlapping Community Structure (AFOCS)

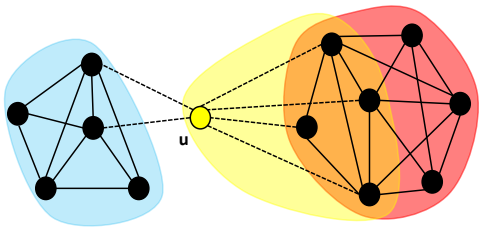


Figure 6: When a new node u is introduced, u could gather some nodes from an existing community (red) to form a new community (yellow)

initially requires a basic community structure $\mathcal{C}_0$. To obtain this basic structure, we apply $DOCA$ algorithm (Alg. 1, Alg. 2) at the first network snapshot, i.e. we execute $DOCA$ on the network G_0 and then let $AFOCS$ adaptively handle this structure as the network evolves.

Handling a new node. Let us discuss the first case when a new node u and its associated links are introduced to the network. Possibilities are (1) u may come with no adjacent edge or (2) with many of them connecting one or more possibly overlapped communities. If u has no adjacent edge, we simply join u in the set of outliers and preserve the current community structure.

The interesting case happens, and it usually does, when u comes with multiple links connecting one ore more existing communities. Since network communities can overlap each other, we need to determine which ones u should join in in order to maximize the gained internal density. But how can we quickly and effectively do so? By Lemma 2, we give a necessary condition for a new node in order to join in an existing community, i.e. our algorithm will join node u in C if the number of connections u has to C suffices: $d_{ui} > \frac{2|C^{in}|}{|C|-1}$. However, failing to satisfy this condition does not necessarily imply that u should not belong to C, since it can potentially gather some substructure of C to form a new community (Figure 6). Thus, we also need to handle this possibility. Alg. 4 presents the algorithm.

Lemma 2 *Suppose u is a newly introduced node with d_{ui} connections to each adjacent community C_i. u will join in C_i if $d_{ui} > \frac{2|C_i^{in}|}{|C_i|-1}$.*

The analysis of Alg. 4 is shown by Lemma 3. In particular, we show that this procedure achieves at least 0.83% internal density of the optimal assignment for u, given the prior community structure.

Lemma 3 *Alg. 4 produces a community assignment that, prior to the community combination process, achieves $\Psi(\mathcal{C}_t) \geq \tau(4) \times \Psi(OPT(u)_t)$ where $OPT(u)_t$ is the optimal community assignment for u at time t, given the prior community structure $\mathcal{C}_{t-1}$.*

Handling a new edge: In case where a new edge $e = (u, v)$ connecting two existing vertices u and v is introduced, we divide it further into two four smaller cases: (1) e is solely inside a single community C (2) e is within the intersection of two (or more) communities (3) e is joining two separated communities and (4) e is crossing overlapped communities. If e is totally inside a

Algorithm 4: Handling a new node u

Input : The current community structure $\mathcal{C}_{t-1}$
Output: An updated structure $\mathcal{C}_t$.

$C_1, C_2, ..., C_k \leftarrow$ Adjacent communities of u;
for $i = 1$ *to* k **do**
    if $d_{ui} > \frac{2|C_i^{in}|}{|C_i|-1}$ **then**
        $C_i \leftarrow C_i \cup \{u\}$;
    end
    else
        $C \leftarrow N(u) \cap C_i$;
        if $\Psi(C) \geq \tau(C)$ *and* $|C| \geq 4$ **then**
            $C_i \leftarrow C_i \cup \{u\}$;
        end
    end
end
/*Checking new community from outliers*/;
for $v \in C_0$ *and* $Com(v) \cap Com(u) = \emptyset$ **do**
    $C \equiv N(u) \cap N(v)$;
    if $\Psi(C) \geq \tau(C)$ *and* $|C| > 4$ **then**
        Define C a new community;
    end
end
Merging overlapping communities on $C_1, C_2, ..., C_k$ and C_0;
Update $\mathcal{C}_t$;

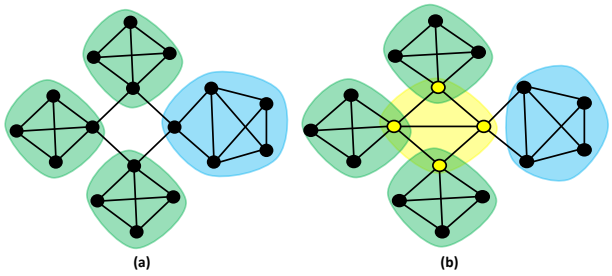


Figure 7: (a) The network with 4 disjoint communities (b) When the central edge is added, the central nodes form a new community (yellow)

community C, its presence will strengthen C's internal density and by Lemma 4, we know that adding e should not split the current community C into smaller substructures. The same reaction applies in the second subcase when e is within the intersection of two communities since their inner densities are both increased. Thus, in these first two cases, we leave the current network structure intact.

Algorithm 5: Handling a new edge (u, v)

Input : The current community structure $\mathcal{C}_{t-1}$.
Output: An updated community structure $\mathcal{C}_t$.

if $((u, v) \in$ *a single community OR* $(u, v) \in C_u \cap C_v)$ **then**
  $\mathcal{C}_t \leftarrow \mathcal{C}_{t-1}$;
end
else if $Com(u) \cap Com(v) = \emptyset$ **then**
  $C \leftarrow N(u) \cap N(v)$;
  if $\Psi(C) \geq \tau(C)$ **then**
    Define C a new community;
    Check for combining on $Com(u)$, $Com(v)$ and C;
  end
  else
    for $D \in Com(u)$ *(or* $D' \in Com(v)$*)* **do**
      if $\Psi(D \cup \{v\}) \geq \tau(D)$ *(or* $\Psi(D' \cup \{u\}) \geq \tau(D')$*)* **then**
        $D \leftarrow D \cup \{v\}$ (or $D' \leftarrow D' \cup \{u\}$);
      end
    end
    Merging overlapping communities on D's (or D');
  end
  Update $\mathcal{C}_t$;
end

Handling the last two subcases is complicated since any of them can either have no effect on the current network structure or unpredictably form a new network community, and furthermore can overlap or merge with the others (Figure 7). However, there is still a possibility that the introduction

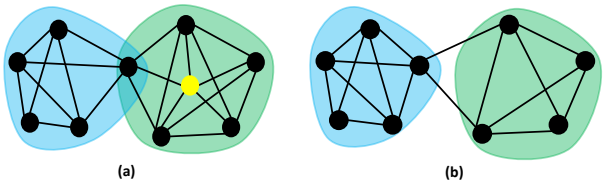


Figure 8: (a) Two overlapped communities (b) When the central node is removed, the new structure consists of two disjoint communities

of this new link, together with some substructure of C_u or C_v, suffices to form a new community that can overlap with not only C_u and C_v but also with some of the others. The other subcases can be handled similarly. Alg. 5 describe this procedure.

Lemma 4 *If an new edge (u, v) is introduced solely inside a community C, it should not split C into smaller substructures.*

Removing an existing node: When an existing node u is about to be removed from the network, all of its adjacent edges will also be removed as a consequence. If u is an outlier, we can simply exclude u and its corresponding links from the current structure and safely keep the network communities unchanged.

In unfortunate situations where u is not an outlier, the problem becomes very challenging in the sense that the resulting community is complicated: it can either be unchanged, or broken into smaller communities, or could probably be merged with the other communities. To give a sense of this effect, let's consider two examples illustrated in Figure 8. In the first example, when C is almost a full clique, the removal of any node will not break it apart. However, if we a remove node that tends to connect the others within a community, the leftover module is broken into a smaller one together with a node that will later be merged to one of its nearby communities. Therefore, identifying the leftover structure of C is a crucial task once a vertex u in C is removed.

Algorithm 6: Removing a node u

Input : The current community structure $\mathcal{C}_{t-1}$.
Output: An updated structure $\mathcal{C}_t$.

for $C \in Com(u)$ *and* $\Psi(C \backslash \{u\}) < \tau(C \backslash \{u\})$ **do**
  $LC \leftarrow$ Local communities by Alg 1 on $C \backslash \{u\}$;
  for $C_i \in LC$ *and* $|C_i| \geq 4$ **do**
    $S_i \leftarrow$ Nodes such that $\Psi(C_i \cup S_i) \geq \tau(C_i \cup S_i)$;
    $C_i \leftarrow C_i \cup S_i$;
  end
  Merging overlapping communities on LC;
end
Update $\mathcal{C}_t$;

To quickly handle this task, we first examine the internal density of C excluding the removed node u. If the number of internal connections still suffices, e.g $\Psi(C\backslash\{u\}) \geq \tau(C\backslash\{u\})$, we can safely keep the current network communities intact. Otherwise, we apply Alg. 1 on the subgraph induced by $C\backslash\{u\}$ to quickly identify the leftover modules in C, and then let these modules hire a set of unassigned nodes $\Psi(C)$ that help them increasing their inner densities. Finally, we locally check for community combination, if any, by using an algorithm similar to Alg. 2.

Removing an edge: In the last situation when an edge $e = (u, v)$ is about to be removed, we divide it further into four subcases similar to those of a new edge (1) e is between two disjoint communities (2) e is inside a sole community (3) e is within the intersection of two (or more) communities and finally (4) e is crossing overlapping communities.

In the first subcase, when e is crossing two disjoint communities, its removal will make the network structure more clear (since we now have less connections between groups), and thus, the current communities should be keep unchanged. When e is totally within a sole community C, handling its removal is complicated since this can lead to an unpredictable transformation of the host module: C could either be unchanged or broken into smaller modules if it contains substructures which are less attractive to each other, as depicted in Figure 9. Therefore, the problem of identify the structure of the remaining module becomes the central part for not only this case but also for the others.

Algorithm 7: Removing an edge (u, v)

Input : The current structure $\mathcal{C}_{t-1}$.
Output: An updated community structure $\mathcal{C}_t$.

if (u, v) *is an isolated edge* **then**
  $\mathcal{C}_t = (\mathcal{C}_{t-1}\backslash\{u, v\}) \cup \{u\} \cup \{v\}$;
end
else if $d_u = 1$ *(or* $d_v = 1$*)* **then**
  $\mathcal{C}_t = (\mathcal{C}_{t-1}\backslash C(u)) \cup \{u\} \cup C(v)$;
end
else if $C \equiv C(u) \cap C(v) = \emptyset$ **then**
  $\mathcal{C}_t = \mathcal{C}_{t-1}$;
end
else if $\Psi(C\backslash(u, v)) < \tau(C\backslash(u, v))$ **then**
  /*Here $C \neq \emptyset$*/;
  $LC \leftarrow$ Local communities by Alg 1 on $C\backslash(u, v)$;
  Define each $L \in LC$ a local community of $\mathcal{C}_{t-1}$;
  Merging overlapping community on L's;
end
Update $\mathcal{C}_t$;

To quickly handle these tasks, we first verify the inner density of the remaining module and, again utilize the local community location method (Alg. 1) to locally identify the leftover substructures. Next, we check for community combination since these structures can possibly overlap with

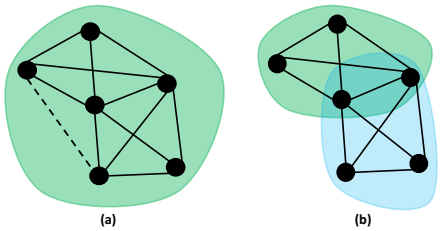


Figure 9: (a) The original community (b) When the dotted edge is removed, the community is broken into two overlapped communities

existing network communities. The detailed procedure is described in Alg. 7.

4 Assessing Network Structure Vulnerability

Impact of Nodes' Failures on Network Components. In this task, we are interested in identifying the set of nodes whose removal triggers a significant restruction of the current community structure. In term of notations, given the input network, the community detection algorithm $\mathcal{A}$ and a positive number k, we formulated the *Community structure Vulnerability Assessment* (CVA) which aims to find a set S of k nodes whose removal maximally transforms the current network community structure to a totally different one, evaluated via the Normalized Mutual Information measure.

Definition 1 *Given a network represented by an undirected and unweighted graph G, a specific community detection algorithm $\mathcal{A}$, and a positive integer $k \leq N$, we seek for a subset $S \subseteq V$ such that $S = \underset{S' \subseteq V, |S'|=k}{\operatorname{argmin}} \{NMI_X(S')\}$, where $X \equiv \mathcal{A}(G)$, and $NMI_X(S') \equiv NMI(X, \mathcal{A}(G[V \backslash S']))$ for any $S' \subseteq V$.*

Our major findings of this tasks are: (1) We analyzed conditions that can possibly lead to the minimization of NMI on community structures. (2) We devised an approximation algorithm for the case $k = 1$, and suggested multiple heuristic algorithms for CVA problem. We validated the effectiveness of our solutions on both synthesized data with known community structures and real-world traces including Arxiv citation network, Facebook, and Foursquare social networks. The details can be found in Alim et al. (WI) and Nguyen et al. (ASONAM).

We have provided the basic results for the NMI analysis as follows:

Lemma 5 *There is a graph $G = (V, E)$ in which $NMI_X()$ is not a submodular function. Moreover, there are subsets $L \subseteq T \subseteq V$ such that $NMI_X(T) \geq NMI_X(L)$ (where L, T are sets of removed nodes).*

Theorem 9 *Given two community assignments $A \subseteq B$, there is $s \notin A, B$ such that $NMI_X(A + x) - NMI_X(A) < NMI_X(B + s) - NMI_X(B)$.*

We provided three algorithms to find a subset S. Our first heuristic algorithm is oriented based on the modularity contributions of network communities in G. There are two versions in general, called $greedyM_N$ and $greedyM_C$, for this heuristic approach with different priorities given to nodes and communities. In $greedyM_N$, all nodes u's in the network are ranked based on their modularity contributions $q_{u,C}$'s, and the top k nodes are selected in the solution set. The second algorithms, $greedyM_C$, consists of two steps: it first finds the community C having the most modularity contribution q_C, and then selects a node u that has the highest modularity portion $q_{u,C}$ in C until k nodes are included in the solution set.

Our second heuristic approach $genEdge$ for CVA problem is based on the component. Basically, given a community structure X and the algorithm $\mathcal{A}$, $genEdge$ tries to find nodes which can potentially break current communities into smaller ones of the relatively same size, where the preference given to large-size communities. In particular, $genEdge$ looks into communities X_i's of X, ordered by their sizes, and selects nodes that can divide this community into more subcomponents.

For any nodes $u, v \in C$, if edge (u, v) is not in E, we call it a missing edge in C. In addition, we call an edge in C "negative" if it is incident to a missing edge in C, and "positive" otherwise. We define the concept of *generating edges* of C as follows.

Definition 2 *(Generating edge) For any edge (u, v) in C, if $C = (C \cap N(u) \cap N(v)) \cup \{u, v\}$ and $\Psi(C) \geq \tau(C)$, we call (u, v) a generating edge of C. We further call C a local core generated by (u, v) and write $gen(u, v) = C$.*

For any community C of G, a set $L \subseteq E$ is called a "generating edge set" of a C if $\cup_{(u,v)\in L} gen(u, v) = C$. Since C can be generated by different generating edge sets and we are constrained on the node budget, we would intuitively seek for the generating edge set of minimal cardinality.

Definition 3 *(The Minimum Generating Edge Set) Given a community C of G, the MGES problem seeks for a generating edge set L^* of C with the smallest cardinality.*

The cores generated by edges in a MGES of a community C of G are tightly connected and they all together compose C. As a result, if we delete an endpoint of every edge in a MGES, C will be broken into smaller modules with the number of modules is at least the number of edges in a MGES (Lemma 6). Since our goal is to break the current community structure X into as many new communities as possible, the removal of crucial nodes defined by edges in a MGES will be a good heuristic for this purpose. But first and foremost, we need to characterize all MGESs in the current community structure X based only on the input network G. Lemma 7 realizes the location of the generating edge(s) of a local core in a community C: they have to be adjacent to nodes with the highest degree in C. Based on this result, we present in Alg. 8 a procedure that can correctly find the MGES of a given community C (Theorem 10).

Lemma 6 *Let L^* be a MGES of a community C. The removal of an endpoint in every edge of L^* will break C into at least $|L^*|$ subcommunities.*

Algorithm 8: An optimal algorithm for finding the MGES

Input : Network $G = (V, E)$ and a community $C \in X$;
Output: Minimum generating edge set L^* of C;

1 Mark all nodes as "unassigned" and $L^* = \emptyset$;
2 Remove all negative edges in C. If any edge(s) survives, they are candidate for generating edges in their corresponding communities, include them to L^*, go to step 3. Else, go to step 4;
3 Reconstruct local cores based on generating edges found in step 2. Mark all nodes in those communities as "assigned". Discard generating edges in L^* that fall into any newly constructed communities. Return if all edges are assigned;
4 Find the set U as in Lemma 7. Find the edge in $NE(U)$ that can generate a local community having the largest size. Include this edge to L^* and mark all nodes in the new local community as "assigned". Ties are broken randomly. Return if all edges are assigned;
5 If there are still unassigned nodes, say the set $I \subseteq C$, construct $G' = G[(I \cup N(I)) \cap C]$. Go back to step 2;

Lemma 7 *Let C be a subset of V, $U = \{u \in C | d_u^C$ is maximum in $C\}$ and $NE(U) = \{(u, v) | u \in U$ or $v \in U$ but not both $\}$. Then, $|NE(U) \cap L^*| \geq 1$.*

Algorithm 9: $genEdge$ - A node selection strategy for CVA based on generating edges

Input : Network $G = (V, E)$, $X = \mathcal{A}(G)$;
Output: A set $S \subseteq V$ of k nodes;

1 Use Alg. 8 to find $L^*_{X_i}$ for all communities X_i's in X;
2 Sort all communities X_i's in X by their sizes of MGESs;
3 Sort all nodes in G by the number of generating edges that they are incident to in X_i. If there is a tie, sort them by their degrees in G;
4 Return top k nodes from step 3;

Theorem 10 *Let d_C be the maximum in-degree of a node in C. Alg. 8 takes $O(d_C|C|)$ time in the worst case scenario and returns an optimal solution for MGES problem.*

Impact of Edges' Failures on Network Components. In this task, we are interested in identifying the set of edges whose removal triggers a significant reconstruction of the current community structure, defined as follows:

Definition 4 *(DBC) Given an undirected graph $G = (V, E)$, and a set $\mathscr{C}$ of k communities, find a subset $S \subset E$ of minimum cardinality such that removing S from the graph breaks every community in $\mathscr{C}$.*

Our major findings of this task are:

- We defined the framework for community structure fragility. At first we introduced the density based broken community (DBC) problem for breaking k communities with the minimum number of edge removals and provided an approximation algorithm, namely CVA, with theoretical performance guarantee, $O(\log k)$. Its pseudo-code is shown in Algorithm 10.
- To analyze the vulnerability of the community structures in a broader sense, we extended the problem formulation to communities produced from an *arbitrary* community detection algorithm. We offered an efficient heuristic to break the communities and identify the set of critical edges.
- We conducted extensive experiments with different parameters to mine interesting observations about the behavior of broken communities after edge removal.

Algorithm 10: CVA: An approximation algorithm for finding the critical edges

Data: Network $G = (V, E)$, $DeletionVector\ D$, $\mathscr{C}$, $|\mathscr{C}| = k$
Result: A set $S \subseteq E$ edges

$S \leftarrow \emptyset$;
$C \leftarrow \emptyset$;
for *each edge* $e \in E$ **do**
    compute the gain $f(e)$;
end
while $|C| < k$ **do**
    $e' \leftarrow \underset{e \in E}{\operatorname{argmax}}\{f(e)\}$;
    In case of a tie, choose randomly;
    $S \leftarrow S \cup \{e'\}$;
    for $l = 1$ *to* k **do**
        if $C_l \notin C$ **then**
            if $e' \in C_l$ **then**
                $D_l \leftarrow D_l - 1$;
                if $D_l \leq 0$ **then**
                    $C \leftarrow C \cup \{C_l\}$;
                    $f(e) = f(e) - 1$ for all $e \in C_l$;
                end
            end
        end
    end
end
return S;

The details can be found in Alim et al. (ASONAM).

For general definition of community structure, we extended the DBC problem to the following one:

Definition 5 *(Broken Community) Consider a community detection algorithm $\mathscr{A}$, which produces a collection $\mathscr{C}$ of communities on graph G (written $\mathscr{C} = \mathscr{A}(G)$). Let G' be a new graph after removal of a set of edges, and let $\mathscr{C}' = \mathscr{A}(G')$. Let $\gamma \in (0, 1)$. A community $C \in \mathscr{C}$ is said to be* broken *in graph G' if there does not exist a community $C' \in \mathscr{C}'$ satisfying*
*(*i*) $C' \subset C$, and (*ii*) $|C'|/|C| > \gamma$*

We have shown that partitioning a community C into at least c ϵ-balanced subparts, where $\gamma c \geq 1 + \epsilon$ makes it broken. Therefore, we developed the following Algorithm 11.

Algorithm 11: CCF: A heuristic algorithm for breaking communities

Data: Network $G = (V, E)$, k Communities $\mathscr{C}$, strictness threshold γ
Result: A set $S \subseteq E$ of edges

```
S ← φ;
c ← z : z is the smallest integer satisfying zγ ≥ 1 + ε;
for each community C_i ∈ 𝒞 do
    compute the c-way balanced partitioning;
    Cut_i = set of edges to cut C_i into c parts;
    S ← S ∪ Cut_i;
end
return S;
```

5 Evacuation Modeling

The research summarized here is from a current submission by Vogiatzis and Pardalos to the *European Journal of Computational Optimization*. Disaster management and evacuation planning are both of utmost importance for the societal welfare of modern countries and states. On top of its humanitarian benefits, proactive planning of a response to a natural or man-made disaster also holds significant cost savings.

Most approaches tackling the evacuation planning problem can be categorized based on their underlying methods. Typically, problems of the sort are formulated and tackled as large-scale linear programming problems, with approaches that generalize network flow algorithms. The problem with such methods stems from the large-scale nature of the problem, which does not enable us to solve real-life scenarios. A second approach utilizes simulation and agent-based methods to stochastically imitate the behaviors of drivers and unexpected delays/failures in the infrastructure. Once more, due to the large-scale nature of the problem, these methods are very computationally expensive, especially with the addition of more realistic constraints. Last, but not least, there exist many approaches in literature that fall within the general spectrum of heuristics, which try to take advantage of specific characteristics of the evacuation process.

In this work we also consider centrality indices within a transportation network. While centrality is not a novel idea (as it has been around since the early 1950s), to the best of our knowledge it has

not been applied on the evacuation modeling field. Node centrality represents the importance, or "criticality", of a given node in the grand scheme of the network, and is considered fundamental in network analysis.

Consider a simple, directed graph $G(V, E)$, with $|V| = n$ vertices, and $|E| = m$ edges. Two vertices i and j are said to be connected if there exists a path of vertices beginning at i and ending in j, i.e. $\{v_0, \ldots, v_k\}$ where $v_0 = i$ and $v_k = l$, and for every two consecutive vertices v_l, v_{l+1} in the path, we have that $(v_l, v_{l+1}) \in E$. A graph G is connected if any pair of nodes $i, j \in V$ are connected: it is a common assumption that transportation networks are always connected. Hence, we also make this assumption for simplicity.

As we are considering a time expanded network, we deal with an instance of the graph G at any given time $t \in T$, where $|T|$ is assumed to be a big enough horizon to cover our evacuation operations. Hence, we have a graph $G^{(t)}$, with its node set, $V^{(t)}$, and edge set, $E^{(t)}$. Every edge is also associated with a parameter m_{ij} that represents the amount of time it would take a vehicle to traverse it. In transportation networks, this can be represented more accurately with a random variable and/or a dependence on the flow currently using the street; however, for our purposes of network decomposition for evacuation problems, it was deemed unnecessary and as such it is treated as a parameter. A subset of nodes $S \subset V$ is described as safe, and is the destination of any vehicle in the transportation network during the evacuation process. We further assume that vehicles do not have a preference on the node $i \in S$ they would like to arrive in.

Moreover, we consider that certain information is readily available on the sets of vertices and edges, before modeling and solving the evacuation problem begins. Namely, each edge $(i, j) \in E^{(t)}$ has a capacity on the number of vehicles it can accommodate at any time $t \in T$, symbolized by $u_{ij}^{(t)}$. For simplicity, it can be generally assumed that $u_{ij}^{(t)} = u_{ij}, \forall t \in T$. The capacity is extended to nodes as well, as we assume that a node $i \in V$ is associated with an upper bound on the number of vehicles that can be there at a given time t, $c_i^{(t)}$. Nodes also have an initial, known demand, $d_i^{(0)}$. Last, we assume that information on the level of danger the node is under at any given time $t \in T$. This can be provided to us by simulation of the underlying disaster. In order to help us with our mathematical formulation, we assume $l_i^{(t)} = 0$ for all $i \in S$; the remaining nodes $i \in V \setminus S$ take a value $l_i^{(t)} > 0$, with larger values signaling the necessity for immediate evacuation.

For betweenness centrality, we use the definition typically encountered in literature, which is shown in (23) for a node $k \in V$, and in (24) for a set of nodes $N \subseteq V$.

$$\mathcal{C}(k) = \sum_{i \in V \setminus \{k\}} \sum_{j \in V \setminus \{k\} : i \neq j} \frac{g_{ij}(k)}{g_{ij}} \tag{23}$$

$$\mathcal{C}(N) = \sum_{i \in V \setminus N} \sum_{j \in V \setminus N : i \neq j} \frac{g_{ij}(N)}{g_{ij}} \tag{24}$$

In the above expressions, $g_{ij}(k)$ is the number of geodesic (shortest) paths connecting nodes i and j that pass through k, whereas $g_{ij}(N)$ the same for group of nodes N. The denominator in both

cases is the total number of geodesic paths connecting nodes i and j, which can be bigger than 1 in the case of multiple alternate shortest paths. Observe that the group betweenness of a set N cannot be calculated by adding the individual node betweenness of each node members.

Let $x_{ij}^{(t)}$ denote the number of vehicles traversing edge $(i,j) \in E^{(t)}$ at time t. For every node $i \in V$, let $d_i^{(t)}$ represent the number of vehicles waiting there at time t. Notice that the demands for the nodes at each time $t > 0$ are treated as variables, which enables us to instruct vehicles to wait at their position instead of moving, whenever the decision to move them would hinder the success of the evacuation plan. For $t = 0$, $d_i^{(0)}$ is treated as an input parameter. Further, we introduce binary variable y_{ij} as follows:

$$y_{ij} = \begin{cases} 1, & \text{if edge } (i,j) \text{ is reversed during the evacuation process,} \\ 0, & \text{otherwise.} \end{cases}$$

From the definition of the set of y variables, it is clear that we consider only the case that a street can be declared reversed (along with the necessary policing and precautions) from the beginning of the evacuation process and cannot change back to its original sense until the end of operations. Last, we assume we are given a horizon of periods $T = \{1, 2, \ldots, |T|\}$, during which time a maximum number of vehicles is to reach a safe node. Equivalently, the number of vehicles still evacuating can be minimized. The mathematical formulation can now be presented in (17)-(34).

$$\min \quad \sum_{t \in T} \sum_{i \in V} l_i^{(t)} d_i^{(t)} + \sum_{t \in T} \sum_{(i,j) \in E^{(t)}} \frac{l_i^{(t)} + l_j^{(t)}}{2} x_{ij}^{(t)} \tag{25}$$

$$s.t. \quad x_{ij}^{(t)} \leq (1 - y_{ij}) u_{ij} + y_{ji} u_{ji}, \qquad \forall (i,j) \in E^{(t)}, \forall t \in T \tag{26}$$

$$y_{ij} + y_{ji} \leq 1, \qquad \forall (i,j) \in E \tag{27}$$

$$d_i^{(t)} \leq c_i, \qquad \forall i \in V^{(t)}, \forall t \in T \setminus \{0\} \tag{28}$$

$$d_i^{(t+1)} = d_i^{(t)} - \sum_{j:(i,j) \in E^{(t)}} x_{ij}^{(t)} + \sum_{\substack{j:(j,i) \in E^{(t-m_{ij})} \\ t - m_{ij} \geq 0}} x_{ji}^{(t-m_{ij})}, \qquad \forall i \in V^{(t)}, \forall t \in T \tag{29}$$

$$\sum_{(i,j) \in E} y_{ij} \leq k, \qquad \forall t \in T \tag{30}$$

$$x_{ij}^{(t)} = 0, \qquad \forall i,j \in V^{(t)} : l_i^{(t)} < l_j^{(t)} \tag{31}$$

$$x_{ij}^{(t)} \geq 0, \qquad \forall (i,j) \in E^{(t)}, \forall t \in T \tag{32}$$

$$d_i^{(t)} \geq 0, \qquad \forall i \in V^{(t)}, \forall t \in T \tag{33}$$

$$y_{ij} \in \{0,1\}, \qquad \forall (i,j) \in E. \tag{34}$$

The objective function in (25) aims to minimize the number of people waiting to be evacuated in the transportation network, either waiting in one of the nodes $i \in V$ or using an edge $(i,j) \in E$. Observe that we employ the simulation-obtained danger factors $l_i^{(t)}$ to derive a weighted sum of

the number of demands that remain to be evacuated; for the edges we use an average estimate of the two endpoints at that specific time $t \in T$. That way, evacuees that are further away would take priority in our operations, as they increase the value of the objective function significantly.

Constraints (26) ensure that whenever we have contraflow present in a segment of the evacuation plan, the number of evacuees using the street can increase over its original capacity, as the capacity of the reverse direction can be employed. Keep note that this is not necessarily double the capacity of the original street, as there exist streets where the capacities are asymmetric per direction. Moreover, when a street is reversed, then the constraint ensures that no flow can be sent employing that same street. Constraint (27) guarantees that in an evacuation setting, at most one of the senses of any street can be reversed at a time. Continuing with the constraints of the problem, in (28) we enforce a capacity at every intersection of the transportation network. That way, we can avoid having evacuees accumulate at any intersection, leading to street capacity losses. It can be assumed that the capacity of the safety nodes is much higher than the capacity of the rest of the nodes in the network.

Constraints (29) are the classical flow preservation constraints, adapted for the time-dynamic network. They enforce that the number of evacuees waiting at an intersection i at a time t are as many as there were in the previous time step $t-1$ when considering also the number of vehicles that are incoming and subtracting the ones outgoing. As a reminder, we have assume that each actual street of the network has been assigned a parameter m_{ij} that stays constant throughout the process which represents the (integer) number of periods it takes to traverse it. Of course, we also need to add a budget constraint, as it is impossible to enable the reversal of every street in our evacuation plan. The budget constraint is shown in (30).

Furthermore, we have constraints (31) that do not allow a vehicle to move from a less dangerous to a more dangerous area. This constraint apart from enabling us with the islanding scheme to be discussed in the following subsection, also guarantees that the objective function is non-increasing from time step to time step. Finally, we guarantee nonnegativity of all variables and the binary nature of the y variables in (32)-(34).

Simple Islanding. The above problem is indeed very hard to solve in real-life, large-scale instances. As an example, it would take a commercial solver multiple hours to obtain an optimal solution, when in fact in many situations, it is imperative that an evacuation plan is devised and implemented much faster than that.

To tackle this issue, we propose a decomposition approach that produces a series of connected clusters, called "islands". For the decomposition, we assume that the number of clusters $|N|$ is given as an input. An initial decomposition approach is presented in Algorithm 12. For this algorithm, it is necessary that we know the distance a vehicle needs to traverse from every node $i \in V \backslash S$ to arrive at a node $j \in S$ belonging to the safe zone, without crossing an edge $(i,j) \in E^{(t)}$ such that $l_i^{(t)} < l_j^{(t+m_{ij})}$. Let this function be referred to as $ModifiedDijkstra(i,S)$, for a given node $i \in V$ and the set of safe nodes in the network, S.

The modification to the original approach to finding all shortest paths from a node to a set of nodes

is pretty straightforward. We perform a preprocessing of the networks $G^{(t)}$ where we remove all edges satisfying $l_i^{(t)} < l_j^{(t+m_{ij})}$; we then proceed to calculate the shortest path from each node to a virtual aggregate node connected to all safe nodes with 0 traversal time. Last, we assume we are given $t_1, t_2, \ldots t_{|N|}$ as threshold distances for a node to belong to cluster $1, 2, \ldots, |N|$, respectively.

Algorithm 12: Decomposition algorithm based on distance from safety and danger factor

```
n ← |N|;
for each node i ∈ V ⊆ S do
    dist(i) ← modifiedDijkstra(i, S);
    for k ∈ {1, 2, ..., n} do
        if dist(i) ≤ t_k then
            N_k ← N_k ∪ {i}
        end
    end
end
```

Betweenness Islanding. An extension of Algorithm 12 is presented in Algorithm 13. This one takes into consideration the betweenness centrality of a cluster C and aims to decompose the central hubs in several, distinct clusters. A simple description of Algorithm 13 can be the following. We first calculate the betweenness of every node in the graph, considering though only the shortest paths from every node $i \in V \setminus S$ to every safe node $s \in S$ (let this procedure be referred to as $SafeBetweenness(i)$ for a node $i \in V$ and $SafeBetweenness(C)$ for a set of connected nodes $C \subseteq V$). We then calculate the group betweenness of all connected clusters N of size n, using only the shortest paths to safety. Note that in regular betweenness calculations, the shortest paths between every pair of nodes (i, j) in the network are considered; instead, here we only consider the shortest paths between any node $i \in V \setminus S$ and any safe zone $s \in S$. The set of size n with the biggest betweenness centrality is selected and its nodes are now to be divided among the different clusters of the islanding scheme. One way for this to be achieved, is to find the shortest path of every other node in the network to the ones found: then each node can be greedily assigned to the one that it can reach the fastest.

The first algorithm is much faster due to the fact that it relies on simple shortest path finding. However, the second algorithm too is tractable for small values of n, as only $\binom{|V|}{n}$ possible set nodes are to be considered.

Experiments can be found in the Vogiatzis and Pardalos submission. They were performed on synthetic networks of varying size (100 to 10,000 nodes) and real-life transportation network instances, where 10% of the nodes are selected to be safe zones. Specifically experiments focus the city of Jacksonville, Florida, which presents a real challenge as it is represented by a huge transportation network. In the Jacksonville network, the safe zones are pre-selected and known, and simulated storms are obtained through the CSEVA package. Figures depicting the algorithms' performances on the Jacksonville network are given in Figure 10 and Figure 11.

Algorithm 13: Decomposition algorithm based on betweenness

$max \leftarrow 0$;
$N_0 \leftarrow \emptyset$;
for *each set of nodes* $N \subseteq V$ **do**
  if $isConnected(N)$ && $|N| = n$ **then**
    $btn = SafeBetweenness(N)$;
    if $max < btn$ **then**
      $max \leftarrow btn$;
      $N_0 \leftarrow N$;
    end
  end
end
for *each node* $i \in V \setminus N_0$ **do**
  $k = argmin\{d_{ik} : k \in N_0\}$;
  $N_k \leftarrow N_k \cup \{i\}$;
end

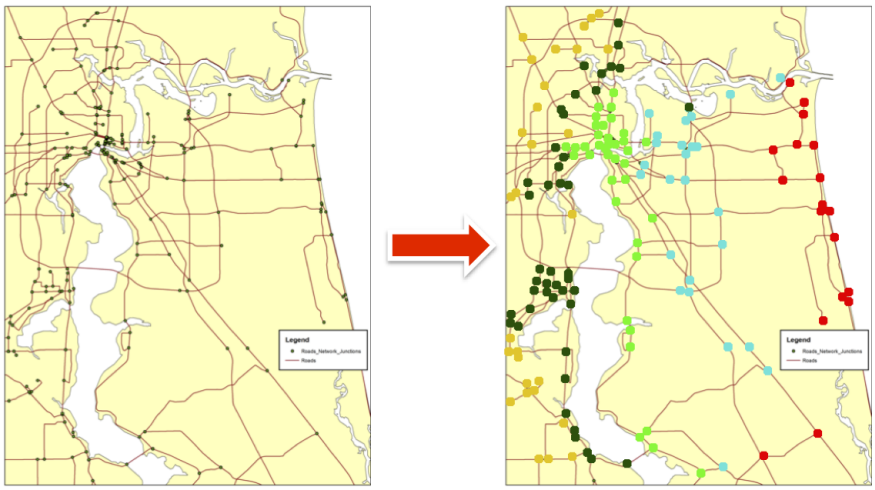


Figure 10: An example of how the first islanding scheme (Algorithm 12) works for the city of Jacksonville, FL.

6 Multidimensional Assignment Problem

This research is published in the *European Journal of Operational Research* (see Walteros et al. (2014)). The multidimensional assignment problem (MAP) aims to minimize the overall cost of assignment when matching elements from $\mathcal{N} = \{N_1, \ldots, N_n\}$ ($n > 2$) disjoint sets of equal size m. It comes as a natural generalization of the two-dimensional Assignment Problem (AP), known to be polynomially solvable. Among all the different generalizations of the MAP, the one considered in this study is the *axial* MAP (hereafter referred to as MAP). In an axial MAP, each element of every set must be assigned to exactly one of m disjoint n-tuples, and each n-tuple must

Figure 11: An example of how the first islanding scheme (Algorithm 13) works for the city of Jacksonville, FL.

contain exactly one element of each set. Contrary to the AP, the MAP is known to be NP-hard for $n > 2$.

The MAP can be modeled as the following integer (0-1) program

$$(MAP): \min \quad \sum_{i_1 \in N_1} \sum_{i_2 = N_2} \cdots \sum_{i_n \in N_n} c_{i_1 i_2 \ldots i_n} x_{i_1 i_2 \ldots i_n} \tag{35}$$

$$s.t. \quad \sum_{i_2 \in N_2} \sum_{i_3 \in N_3} \cdots \sum_{i_n \in N_n} x_{i_1 i_2 \ldots i_n} = 1, \qquad i_1 \in N_1 \tag{36}$$

$$\sum_{i_1 \in N_1} \cdots \sum_{i_{s-1} \in N_{s-1}} \sum_{i_{s+1} \in N_{s+1}} \cdots \sum_{i_n \in N_n} x_{i_1 i_2 \ldots i_n} = 1, \quad i_s \in N_s, \quad s = 2, \ldots, n-1 \tag{37}$$

$$\sum_{i_1 \in N_1} \sum_{i_2 \in N_2} \cdots \sum_{i_{n-1} \in N_{n-1}} x_{i_1 i_2 \ldots i_n} = 1, \qquad i_n \in N_n \tag{38}$$

$$x_{i_1 i_2 \ldots i_n} \in \{0, 1\}, \qquad i_s \in N_s, \quad s = 1 \ldots n, \tag{39}$$

where, for every n-tuple $(i_1, i_2, \ldots, i_n) \in N_1 \times N_2 \times \cdots \times N_n$, variable $x_{i_1 i_2 \ldots i_n}$ takes the value of one if elements of the given n-tuple belong to the same assignment, and zero otherwise. The total assignment cost (35) is computed as the cost of matching elements from different sets together. As an example, an assignment which selects elements $(i_1, i_2, \ldots, i_n)$ to be grouped together would have a cost of $c_{i_1 i_2 \ldots i_n}$.

Depending on the definition of the assignment costs, there are several variations of the MAP that can be considered. These variations are mainly associated with cases where the assignment cost of each n-tuple can be decomposed as a function of all possible pairwise assignment costs between

elements of different sets. That is, $c_{i_1 i_2 \ldots i_n} = f(c_{i_1 i_2}, \ldots, c_{i_n i_{n-1}})$, where $f : N_1 \times N_2 \cup \cdots \cup N_{n-1} \times N_n \to \mathbb{R}$ and $c_{i_s i_t}$ is the cost of assigning together elements $i_s \in N_s$ and $i_t \in N_t$, for $s \neq t$. In general, the main advantage of having decomposable cost functions is that there may be ways of tackling the problem without having to completely enumerate all of the different assignment costs, which can be exponentially many. Moreover, most of these MAP variations can be associated with a weighted n-partite graph, in which the elements are represented by the vertices of the graph, each of the edges describes the decision of assigning two elements within the same n-tuple, and the weights on the edges account for the corresponding assignment costs. In particular, **we consider the case where each n-tuple of any feasible assignment is assumed to form a star.**

For the case of the stars, one element of each tuple is assigned to be a center (or representative) and the other elements are considered to be the leafs (or legs) of the star. Note that, contrary to the case of the cliques, each tuple can generate many different star configurations, depending on which element is selected as the center. Assuming for example, that the center is element i_s, the cost of the induced star is the sum of the pairwise costs between i_s and the other elements of the tuple. In view of these multiple possible configurations, the cost of tuple $(i_1, i_2, \ldots, i_n) \in N_1 \times N_2 \times \cdots \times N_n$ is defined as the minimum cost among the costs of all the possible star configurations of the tuple. That is,

$$c_{i_1 i_2 \ldots i_n} = \min_{i_s \in \{i_1, i_2, \ldots, i_n\}} \left\{ \sum_{t \in \{1,2,\ldots,n\} \setminus \{s\}} c_{i_s i_t} \right\} \tag{40}$$

The aforementioned MAP version is called the *multidimensional star assignment problem* (MSAP), because of the particular structure that each feasible assignment has. There are several contexts in which using star costs can prove beneficial when solving multi-sensor multi-target tracking problems. In particular, when the assignment costs have metric properties (i.e, nonnegativity, symmetry, and subadditivity), it is interesting to see that in some cases, considering all pairwise costs within the assignments (e.g., the clique case) is not necessary to obtain a valid solution. We provide two examples with costs that satisfy those properties and show that the star cost variation is a valuable tool to solve those problems. We would like to emphasize though, that the star costs are not always a better or worse option compared to other versions such as the clique costs. Instead, it can be seen as an alternative that also provides additional information, and that can contribute to a better analysis depending on the context.

In the first example we aim to identify a set of land mines that are planted on a field. To find the location of the mines, a drone is sent to fly over the field emitting a signal. Once the signal reaches each of the mines, it bounces back and is analyzed by the sensors of the drone. After the drone has flown over the field a number of times and its sensors have collected the set of different signals (several of those associated with each of the mines), it is possible to calculate a set of estimated locations where the mines could be located. The idea behind solving a MAP is to associate the locations that are close to each other, which would help pinpoint the actual positions of the mines. In this context, the assignment costs represent the Euclidean distances between the estimated locations and, since the costs satisfy the triangle inequality, not all the costs need to be considered to obtain a valid association.

Moreover, the expected position of the mine is often considered to be a concurrency point asso-

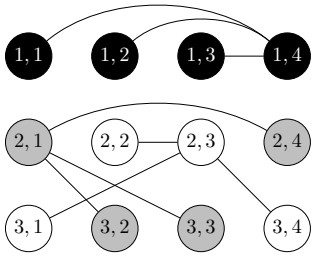


Figure 12: A valid star assignment for a graph with $n = 4$ and $m = 3$

ciated with the positions that are in the same n-tuple, generally the centroid. On the other hand, if star costs are used, the region around the center of the stars can be seen as the zone where the mines are most likely to be located. Under the triangle inequality assumption, it is easy to prove that many of the triangle concurrency points (e.g., the centroid, the incenter, or the circumcenter) are always closer to the center of the star.

The second example is a case where the costs do not represent Euclidean distances and hence, the concept of the geometric center does not have a proper interpretation. Assume we have a group of antennas that are placed to intercept a collection of encrypted messages that we want to decode. Because of the interference and the noise of the environment, when the messages are received by the antennas they are somehow disrupted and thus not perfect for decoding. Each of the antennas can potentially receive a different version of each of the messages. The idea is to identify which of the received versions are the ones that most closely resemble the correct (transmitted) ones, so they can be sent for decoding. Here, since there is no way to compare the disrupted versions with the correct messages, the decision of which versions are finally sent to decode should be made by analyzing the dissimilarities between the received messages. In many contexts like this, the decoding process could be rather difficult. Hence, it is desired to select only one of the disrupted versions of each of the messages. Assuming that the messages are encrypted in some kind of alphabet (e.g., binary codes), using the centroid of the disrupted versions is not a valid approach here. In this framework, the use of the star costs could be of benefit, because the messages associated with the centers of the stars can be the ones selected for decoding (representative). For this problem, the costs can be assumed to be the number of different characters between the messages, the sum of how far apart in the alphabet the different characters of the messages are, or any desired correlation metric.

Given a collection $N_1, \ldots, N_n$ of n disjoint node sets of equal size m and the corresponding collection of pairwise arc sets $E_{12}, \ldots E_{n-1,n}$, where $E_{st} = N_s \times N_t$ for $s < t$, let $\mathcal{G} = (\mathcal{N}, \mathcal{E})$ be a complete n–partite graph, where $\mathcal{N} = \bigcup_{s=1}^{n} N_s$ and $\mathcal{E} = \bigcup_{s<t} E_{st}$. We refer to the ith node of set N_s with the duple (i, s) and use the quadruple (i, s, j, t), to represent the edge between nodes (i, s) and (j, t). Also, let c_{ij}^{st} be a cost value associated with edge (i, s, j, t) that accounts for the cost of assigning nodes (i, s) and (j, t) to the same star. Let a valid star be defined as a subgraph of $\mathcal{G}$ such that, (1) it contains exactly one node from each set $N_1, \ldots, N_n$; (2) one of the nodes (the center) is connected to all of the other nodes of the star (the leafs); and (3) there are no connections between any pair of leafs. Then, the MSAP aims for a set of m disjoint valid stars that cover all the nodes in $\mathcal{G}$. An example of a valid star assignment of an instance, where $n = 4$ and $m = 3$ is given in Figure 12. The three valid stars are colored gray, black, and white.

Continuous nonlinear formulation. Let variables $\mathbf{z}$ and $\mathbf{x}$ be defined as follows:

$$z_i^s = \begin{cases} 1, & \text{if node } (i,s) \in \mathcal{N} \text{ is a star center} \\ 0, & \text{otherwise,} \end{cases}$$

and

$$x_{ij}^{st} = \begin{cases} 1, & \text{if nodes } (i,s) \text{ and } (j,t) \in \mathcal{N} \text{ belong to the same star} \\ 0, & \text{otherwise.} \end{cases}$$

Observe that $x_{ij}^{st} = x_{ji}^{ts}$ for all (i,s,j,t), and $x_{ij}^{ss} = 0$, since $(i,s,j,s) \notin \mathcal{E}$. The initial formulation is presented in (41)-(48), where the objective function (41) aims to minimize the overall assignment cost. Constraints (42) ensure that, if node (i,s) is a center, it must be connected to $(n-1)$ nodes. Conversely, if it is not a center, then it should be connected to one node. Constraints (43)–(44) guarantee that if node (i,s) is a center, it must be connected to exactly one node from each set N_t, for all $t \neq s$. Further, constraints (45) enforce that there are exactly m stars in the optimal solution. Nonlinear constraints (46) guarantee that each node $(i,s) \in \mathcal{N}$ is either connected to a center, or is a center itself, and hence, it can only be connected to leafs. Last, (47)-(48) define the domain of variables $\mathbf{z}$ and $\mathbf{x}$.

$$(INLP): \min \sum_{i=1}^{m}\sum_{j=1}^{m}\sum_{s=1}^{n}\sum_{\{t=1,\ldots,n: s<t\}} c_{ij}^{st} x_{ij}^{st} \tag{41}$$

$$\text{s.t.} \sum_{j=1}^{m}\sum_{t=1}^{n} x_{ij}^{st} = 1 + (n-2) z_i^s, \qquad \forall i = 1,\ldots,m \quad s = 1,\ldots,n \tag{42}$$

$$\sum_{j=1}^{m} x_{ij}^{st} \leq 1, \qquad \forall i = 1,\ldots,m \quad s,t = 1,\ldots,n \tag{43}$$

$$\sum_{j=1}^{m} x_{ij}^{st} \geq z_i^s, \qquad \forall i = 1,\ldots,m \quad s,t = 1,\ldots,n \tag{44}$$

$$\sum_{i=1}^{m}\sum_{s=1}^{n} z_i^s = m \tag{45}$$

$$z_i^s + \sum_{j=1}^{m}\sum_{t=1}^{n} x_{ij}^{st} z_j^t = 1, \qquad \forall i = 1,\ldots,m \quad s = 1,\ldots,n \tag{46}$$

$$z_i^s \in \{0,1\}, \qquad \forall i = 1,\ldots,m \quad s = 1,\ldots,n \tag{47}$$

$$x_{ij}^{st} \in \{0,1\}, \qquad \forall i = 1,\ldots,m \quad s = 1,\ldots,n. \tag{48}$$

Let $RNLP$ be the continuous relaxation of formulation (41)–(48). That is, both $\mathbf{z}$ and $\mathbf{x}$ are allowed to take fractional values between zero and one. We now proceed to prove that in any feasible solution of $RNLP$, all $\mathbf{z}$ variables take integer values. For this proof, assume we have a feasible solution $(\mathbf{z}, \mathbf{x})$. We refer to node (i,s) as white if $z_i^s = 0$, black if $z_i^s = 1$, or gray if $z_i^s \in (0,1)$.

Lemma 8 *In any feasible solution of $RNLP$, a white node can only be connected to black nodes.*

Lemma 9 *In any feasible solution of $RNLP$, a black node can only be connected to white nodes.*

Based on Lemmata 8 and 9, we can deduce that, if there exist gray nodes, then they are only connected to other gray nodes.

Lemma 10 *If there exist gray nodes in a feasible solution, then the number of those is a multiple of n.*

Theorem 11 *In any feasible solution of the $RNLP$, all* $\mathbf{z}$ *variables are binary (i.e., there cannot exist gray nodes).*

Contrary to the case of the $\mathbf{z}$ variables, it is easy to see that there can be feasible solutions where some of the $\mathbf{x}$ variables are fractional. The following theorem states that if there exists an optimal solution where the $\mathbf{x}$ are fractional, there is also an alternative integral solution.

Theorem 12 *If $RNLP$ is feasible, then there always exists an optimal solution of $RNLP$, where all* $\mathbf{x}$ *variables take integer values.*

As a result from Theorems 11 and 12, constraints (47) and (48) can be relaxed in $INLP$, leaving the *c*ontinuous nonlinear optimization problem $RNLP$, henceforth referred to only as NLP. To solve this formulation we can use any available optimizer that handles nonlinear programs. Although, in spite of having a continuous formulation, rather than an integer one, nonlinear constraints (46) still pose a difficult challenge because of their non-convex nature. As an alternative approach, instead of solving NLP, we propose using a standard linearization technique that involves introducing additional variables to replace the bilinear terms in constraints (46). Furthermore, from the results presented in Theorems 11 and 12, we derive additional valid inequalities that strengthen the proposed linear formulation. A description of the linearization and the valid inequalities follows.

The bilinear terms of constraints (46) represent the greatest difficulty of the NLP formulation. Thus, we apply a standard linearization technique by replacing those terms with additional variables ($\mathbf{w}$). Unfortunately, by relaxing the nonlinear constraints, we lose the integrality properties described in Theorems 11 and 12. Hence, the resulting formulation is a *mixed integer linear program* (MIP). On top of that, this reformulation comes with an overhead of $O(n^2m^2)$ variables

and constraints. The proposed reformulation follows:

$$(MIP): \min \sum_{i=1}^{m}\sum_{j=1}^{m}\sum_{s=1}^{n}\sum_{\{t=1,\ldots,n:s<t\}} c_{ij}^{st}x_{ij}^{st} \tag{49}$$

$$\text{s.t. } (42)-(45) \tag{50}$$

$$w_{ij}^{st} \le z_j^t, \qquad \forall i,j=1,\ldots,m \quad s,t=1,\ldots,n \tag{51}$$

$$w_{ij}^{st} \le x_{ij}^{st}, \qquad \forall i,j=1,\ldots,m \quad s,t=1,\ldots,n \tag{52}$$

$$w_{ij}^{st} \ge z_j^t + x_{ij}^{st} - 1, \qquad \forall i,j=1,\ldots,m \quad s,t=1,\ldots,n \tag{53}$$

$$z_i^s + \sum_{j=1}^{m}\sum_{t=1}^{n} w_{ij}^{st} = 1, \qquad \forall i=1,\ldots,m \quad s=1,\ldots,n \tag{54}$$

$$w_{ij}^{st} \in [0,1], \qquad \forall i=1,\ldots,m \quad s=1,\ldots,n \tag{55}$$

$$x_{ij}^{st} \in [0,1], \qquad \forall i=1,\ldots,m \quad s=1,\ldots,n \tag{56}$$

$$z_i^s \in \{0,1\}, \qquad \forall i=1,\ldots,m \quad s=1,\ldots,n, \tag{57}$$

where variable w_{ij}^{st} represents the bilinear term $x_{ij}^{st}z_j^t$, for all $(i,s,j,t) \in \mathcal{E}$, and constraints (51)–(53) are the linearization inequalities. Note that, contrary to the case of the $\mathbf{x}$ variables, $w_{ij}^{st} = x_{ij}^{st}z_j^t \neq x_{ij}^{st}z_i^s = w_{ji}^{ts}$. Thus, both variables w_{ij}^{st} and w_{ji}^{ts} must be included. Finally, using a similar argument as in Theorem 12, it is easy to see that in the above formulation we can relax the integrality constraints of variables $\mathbf{x}$ and $\mathbf{w}$.

Valid Inequalities. In this subsection, we introduce five families of valid inequalities that strengthen the MIP formulation. Since we no longer have the nonlinear set of constraints (46), it is possible (and likely) that the linear relaxation of this formulation produces fractional solutions (i.e., gray nodes). To cope with this issue, we can use the results from Lemmata 8 and 9 to define the following inequalities.

Proposition 1 *For any edge $(i,s,j,t) \in \mathcal{E}$, the inequality $x_{ij}^{st} \le 2 - z_i^s - z_j^t$ is valid.*

This inequality comes from the fact that in any feasible solution of the MIP there are no connections between black nodes (star centers). Clearly, if both variables z_i^s and z_j^t take the value of one, the corresponding variable x_{ij}^{st} cannot be positive.

Proposition 2 *For any edge $(i,s,j,t) \in \mathcal{E}$, the inequality $x_{ij}^{st} \le z_i^s + z_j^t$ is valid.*

Similarly as before (cf. Proposition 1), in any feasible solution of the MIP there are no connections between white nodes (leafs). Thus, if both variables z_i^s and z_j^t are zero, the corresponding variable x_{ij}^{st} must also be zero. If the foregoing constraints are included in formulation (49)–(57), it is possible to relax constraints (54), and therefore remove the extra $\mathbf{w}$ variables. The resulting formulation, referred to as $MIPa$, not only has less variables and constraints than MIP, but also produces a better dual bound.

$$(MIPa): \min \sum_{i=1}^{m}\sum_{j=1}^{m}\sum_{s=1}^{n}\sum_{\{t=1,\ldots,n:s<t\}} c_{ij}^{st}x_{ij}^{st} \tag{58}$$

$$\text{s.t. } (42)-(45) \tag{59}$$

$$x_{ij}^{st} \leq z_i^s + z_j^t, \qquad \forall i,j=1,\ldots,m \quad s,t=1,\ldots,n \tag{60}$$

$$x_{ij}^{st} \leq 2 - z_i^s - z_j^t, \qquad \forall i,j=1,\ldots,m \quad s,t=1,\ldots,n \tag{61}$$

$$z_i^s \in \{0,1\}, \qquad \forall i=1,\ldots,m \quad s=1,\ldots,n \tag{62}$$

$$x_{ij}^{st} \in \{0,1\}, \qquad \forall i=1,\ldots,m \quad s=1,\ldots,n. \tag{63}$$

Theorem 13 *$MIPa$ is a valid formulation for the MSAP.*

We now describe three additional families of valid inequalities.

Proposition 3 *For any node $(i,s) \in \mathcal{N}$, the inequality*

$$z_i^s + \sum_{j=1}^{m}\sum_{t=1}^{n} q_{ij}^{st} \geq 1, \tag{64}$$

where $q_{ij}^{st} \in \{x_{ij}^{st}, z_j^t\}$ is valid.

We refer to this family as the *minimum sum* inequalities. Note that, for each node (i,s), the number of inequalities of this family is $O(2^{mn})$. Thus, we propose the separation procedure presented in Algorithm 14. Given a fractional solution $(\bar{\mathbf{z}}, \bar{\mathbf{x}})$, for each node (i,s), we search for the inequality that is violated the most. For this purpose, for each (j,t) such that $(i,s,j,t) \in \mathcal{E}$, we select as q_{ij}^{st} in (64) the variable associated with the minimum value between $\bar{z}_j^t$ and $\bar{x}_{ij}^{st}$. We repeat this procedure for all $(i,s) \in \mathcal{N}$, adding all violated inequalities.

For the last two families of valid inequalities, consider the fractional solution depicted in Figure 13. This is an optimal solution of the linear relaxation of $MIPa$. First, observe the 4-cycle formed by nodes $(2,1),(1,3),(1,4),(2,2)$. Clearly, in a feasible integral solution the number of arcs within the cycle cannot be greater than two. We analyze this inequality in Proposition 4.

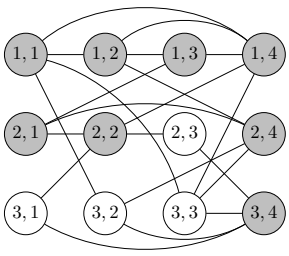


Figure 13: Example of a fractional solution of the linear relaxation of $MIPa$

Algorithm 14: `minSumSeparation`$(\mathcal{G}, i, s, \bar{\mathbf{z}}, \bar{\mathbf{x}})$

$sum = \bar{z}_i^s$;
$cut = z_i^s$;
for (j,t) *such that* $(i,s,j,t) \in \mathcal{E}$ **do**
  if $\bar{z}_j^t < \bar{x}_{ij}^{st}$ **then**
    $sum \leftarrow sum + \bar{z}_j^t$;
    $cut \leftarrow cut + z_j^t$;
  end
  else
    $sum \leftarrow sum + \bar{x}_{ij}^{st}$;
    $cut \leftarrow cut + x_{ij}^{st}$;
  end
end
if $sum < 1$ **then**
  return $cut \geq 1$
end
else
  return $null$
end

Proposition 4 *For any 4–cycle involving nodes* $(i,s),(j,t),(k,u)$*, and* (l,v)*,* $x_{ij}^{st} + x_{jk}^{tu} + x_{kl}^{uv} + x_{il}^{sv} \leq 2$ *is a valid inequality.*

We refer to this family as the *4-cycle* inequalities. Since the number of inequalities of this family is $O(m^4 n^4)$, we can find all violations by total enumeration.

Finally, observe the triplet formed by nodes $(1,3),(2,1)$, and $(2,2)$ in Figure 13. It is easy to see that all three nodes cannot be black and simultaneously have any connection between them. We analyze this inequality in Proposition 5.

Proposition 5 *For any triplet of nodes* $(i,s),(j,t),(k,u)$*, where node* (i,s) *is the center of the triplet,* $x_{ij}^{st} + x_{ik}^{su} + z_i^s + z_j^t + z_k^u \leq 3$ *is a valid inequality.*

We described the above inequality for the case where node (i,s) is the center of the triplet. Although, note that it is possible to obtain two alternative inequalities by choosing any of the other two nodes as the center. We refer to this family as the *triplet* inequalities and, since the total number of possible triplets is $O(n^3 m^3)$, we can find violations by total enumeration. Finally, note that this family of inequalities can be easily generalized for k–tuples, where $k > 3$. However, in practice they are less effective and harder to evaluate, as the total number of k-tuples if $O(n^k m^k)$.

7 Node Deletion

Consider an undirected simple graph $G(\mathcal{V}, \mathcal{E})$ having node set $\mathcal{V} = \{1, \ldots, n\}$, and edge set $\mathcal{E} \subseteq \{(i, j) : i,\ j$ are distinct nodes in $\mathcal{V}\}$. Shen et al. (2012) examine the problem of maximally disconnecting G by deleting a subset of no more than $B < |\mathcal{V}|$ nodes (and all of their incident edges). Define a (maximal) *connected component* as a subgraph such that every pair of nodes in the subgraph is connected by a path, and no path exists between a node outside the subgraph and a node belonging to the subgraph. Another study that we conducted as a part of this grant can be found in Shen and Smith (Networks, 2013). We consider the following network-connectivity metrics, which we apply to G after some subset of nodes has been deleted from it: (i) the number of components in G, (ii) the largest component size in G, and (iii) the minimum cost to reconnect the graph after node deletions, given a set of edge construction costs. We refer to these three problems as MaxNum, MinMaxC, and MaxMinLR, respectively. If there exist alternative optima for these problems, we define an optimal solution as one requiring the fewest node deletions.

An intuitive approach to solving the problems we consider is to greedily delete a node having the largest degree in the current graph along with its adjacent edges, and reiterate until B nodes have been deleted. However, this algorithm does not generally yield even a constant-factor polynomial-time approximation scheme. Figure 14 depicts a MaxNum instance with $B = 1$, in which the greedy algorithm removes the gray node, resulting in one component (excluding the deleted node). However, the optimal solution removes the black node, yielding four components. (Note that by increasing the number of leaf nodes adjacent to the black node, and the number of non-leaf nodes adjacent to the gray node, we can arbitrarily increase the optimality gap between the heuristic and optimal objective function values.) Figure 15 depicts a MinMaxC instance with $B = 1$, where the greedy algorithm deletes the highest-degree node (colored gray), leaving a largest component that has 16 nodes. However, the optimal solution deletes the black node, which results in a smaller largest-component size of five; again, this optimality gap can be made arbitrarily large by suitably expanding the graph. By contrast, we explore an exact optimization algorithm for these problems. Our central approach for these problems is inspired by bilevel optimization techniques used in network interdiction models. We formulate MaxNum, MinMaxC, and MaxMinLR on general graphs as two-stage network interdiction models. We then transform each model into an integrated mixed integer program (MIP). For MaxNum and MinMaxC, we employ results for tree structures in the Shen and Smith (Networks, 2013) paper, and derive a class of valid inequalities designed to improve the solvability of our MIPs, based on polynomial-time DP solutions of subgraphs derived from G.

Complexity Analysis and MIP Formulations. We now formulate the three node deletion problems as bilevel min-max (or max-min) programs, and prove that each is strongly $\mathcal{NP}$-hard. We then demonstrate how to obtain an integrated MIP model for each node deletion problem.

MaxNum. We begin by considering MaxNum, which maximizes the number of components after deleting a subset of nodes. Denote as $\overline{\text{MaxNum}}$ the decision version of MaxNum, which seeks to delete a subset of no more than B nodes, such that the number of components in G is at

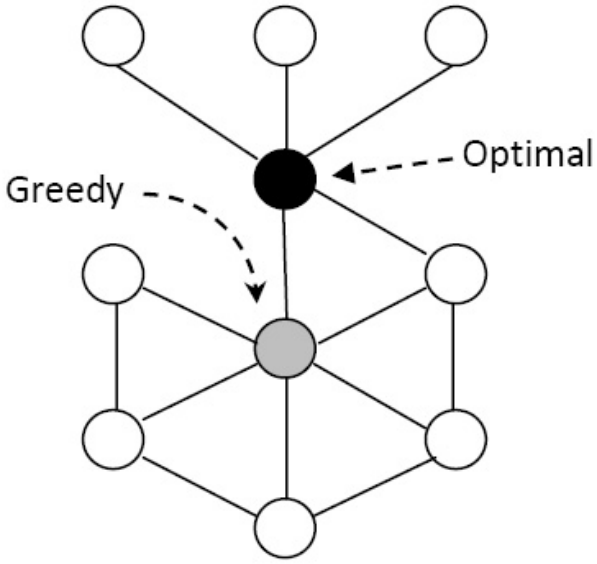


Figure 14: Suboptimality of the Greedy Algorithm in MaxNum for $B = 1$

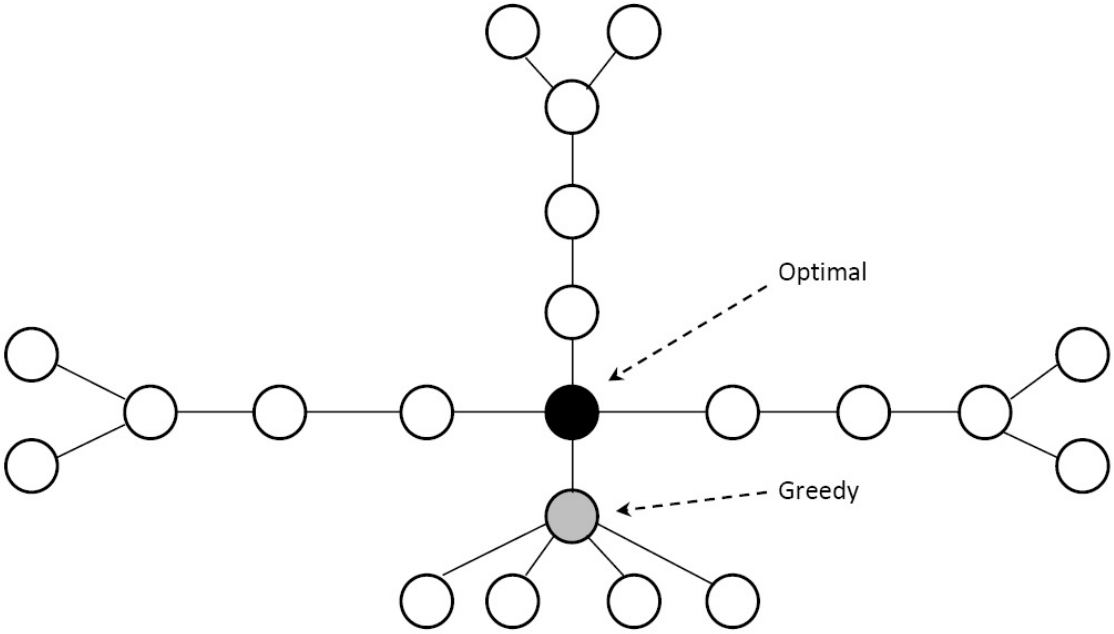


Figure 15: Suboptimality of the Greedy Algorithm in MinMaxC for $B = 1$

least some given integer target value $\mathcal{T}$. All proofs are contained in the published paper.

Theorem 14 *$\overline{\text{MaxNum}}$ is $\mathcal{NP}$-complete in the strong sense.*

Now, to formulate MaxNum as an MIP, define binary variables $x_i,\ \forall i \in \mathcal{V}$, such that $x_i = 1$ if node i is *not* deleted, and $x_i = 0$ otherwise. Also, define $y_{ij} \in \{0,1\},\ \forall (i,j) \in \mathcal{E}$, such that $y_{ij} = 0$ if edge $(i,j) \in \mathcal{E}$ is deleted (due to the deletions of nodes i, j, or both), and $y_{ij} = 1$ otherwise. Note that $y_{ij} = x_i x_j$, i.e., an edge is not deleted if and only if both of its incident nodes are not deleted. Let $\eta(x,y)$ be the number of components remaining in G given a deletion solution (x,y). We give MaxNum as

$$\text{max} \quad \eta(x,y) - \frac{1}{n}\sum_{i\in\mathcal{V}}(1-x_i) \tag{65a}$$

$$\text{s.t.} \quad \sum_{i\in\mathcal{V}}(1-x_i) \le B \tag{65b}$$

$$x_i + x_j - 1 \le y_{ij},\ y_{ij} \le x_i,\ y_{ij} \le x_j \quad \forall (i,j) \in \mathcal{E} \tag{65c}$$

$$x_i \in \{0,1\} \quad \forall i \in \mathcal{V} \tag{65d}$$

$$0 \le y_{ij} \le 1 \quad \forall (i,j) \in \mathcal{E}. \tag{65e}$$

In the objective (65a), we introduce a penalty term $-1/n\sum_{i\in\mathcal{V}}(1-x_i)$ such that if there exists more than one solution that maximizes the number of components, a solution having the fewest deleted nodes is chosen as an optimal solution. (Note that $0 < 1/n\sum_{i\in\mathcal{V}}(1-x_i) \le 1$ for any solution values of x, and that the function $\eta(x,y)$ takes only integer values. Hence, the introduction of this penalty term will not generate suboptimal solutions.) Constraint (65b) limits the number of deleted nodes to be no more than B, and constraints (65c) force $y_{ij} = 1$ if $x_i = x_j = 1$, for all $(i,j) \in \mathcal{E}$. Otherwise, if $x_i = 0$ or $x_j = 0$, y_{ij} will take on a value of 0.

We next formulate the problem of calculating $\eta(x,y)$ using an MIP model on an auxiliary network, and show that the linear programming (LP) relaxation of this formulation yields a convex hull representation of the problem, given that all y-values are binary-valued.

Let $\widetilde{G}(\mathcal{V}\cup\{0\},\mathcal{A})$ denote a transformed directed network, where node 0 will act as a dummy source node. Set $\mathcal{A}$ consists of two directed arcs (i,j) and (j,i) for all edges $(i,j) \in \mathcal{E}$. Our approach requires that a path must exist in $\widetilde{G}$ from node 0 to each node i, for every $i \in \mathcal{V} : x_i = 1$. Define $\widetilde{\mathcal{V}} = \{i \in \mathcal{V} : x_i = 1\}$ as the set of active nodes. Let $\mathcal{A}^*$ be a minimum-cardinality set of arcs that can be constructed between node 0 and every node in $\mathcal{V}$, such that there exists a path from node 0 to i, $\forall i \in \widetilde{\mathcal{V}}$, using arcs in $\mathcal{A}\cup\mathcal{A}^*$. Then the number of graph components equals $|\mathcal{A}^*|$.

Define $FS(i) = \{j : (i,j) \in \mathcal{A}\}$ as the set of nodes adjacent from node i, and $RS(i) = \{j : (j,i) \in \mathcal{A}\}\cup\{0\}$ as the set of nodes adjacent to node $i,\ \forall i \in \mathcal{V}$. Also, $FS(0) = \mathcal{V}$ and $RS(0) = \emptyset$. We define binary variables z_i for all $i \in \mathcal{V}$, such that $z_i = 1$ if we construct arc $(0,i)$ (i.e., if we will use $(0,i)$ in a path from 0 to some node in $\widetilde{\mathcal{V}}$), and $z_i = 0$ otherwise. The goal is to minimize the number of arcs $(0,i)$ constructed over all $i \in \mathcal{V}$, subject to the restriction that at least one path can be routed from node 0 to every active node $k \in \widetilde{\mathcal{V}}$. We associate a different commodity with each node pair $(0,k)$, $\forall k \in \widetilde{\mathcal{V}}$, and define f_{ijk} as the multi-commodity flow variable on arc $(i,j) \in \mathcal{A}$

that routes a path from node 0 to node $k \in \mathcal{V}$. The MaxNum subproblem is given by

$$\eta(x,y) = \min \quad \sum_{i\in\mathcal{V}} z_i \tag{66a}$$

$$\text{s.t.} \quad \sum_{j\in FS(i)} f_{ijk} - \sum_{j\in RS(i)} f_{jik} = a_i x_k \quad \forall i \in \{0\}\cup\mathcal{V},\ k\in\mathcal{V},\ i\neq k \tag{66b}$$

$$f_{0jk} \le z_j \quad \forall j,\ k\in\mathcal{V} \tag{66c}$$

$$f_{ijk} \le y_{ij} \quad \forall (i,j)\in\mathcal{A},\ k\in\mathcal{V} \tag{66d}$$

$$z_i \in \{0,1\} \quad \forall i\in\mathcal{V} \tag{66e}$$

$$f_{ijk} \ge 0 \quad \forall (i,j)\in\mathcal{A},\ k\in\mathcal{V}. \tag{66f}$$

Constraints (66b) are multi-commodity flow balance constraints, in which we define parameters $a_0 = 1$ and $a_i = 0, \forall i \in \mathcal{V}$, i.e., there exists one unit of flow originating at node 0 and terminating at node k, for each $k \in \widetilde{\mathcal{V}}$. Note that constraints (66b) do not require any flow to reach node $k \in \mathcal{V}$ if it is not active (i.e., if $x_k = 0$, then $f_{ijk} = 0,\ \forall (i,j) \in \mathcal{A}$, and $f_{0ik} = 0,\ \forall i \in \mathcal{V}$, is feasible). Constraints (66c) indicate that no flow is permitted on arc $(0,j),\ \forall j \in \mathcal{V}$, if the arc is not constructed, and constraints (66d) prevent flow on arc $(i,j) \in \mathcal{A}$ if it is deleted (where $y_{ij} \equiv y_{ji},\ \forall (i,j) \in \mathcal{A}$). Constraints (66e) and (66f) require z to be binary, and f to be nonnegative, respectively. Formulation (66) can be extended for solving the case where deleted nodes are considered as singleton components, by simply changing the right-hand-sides (RHS) of constraints (66b) to $a_i, \forall i \in \{0\}\cup\mathcal{V}$.

Next, we show that solving (66) is equivalent to solving its LP relaxation.

Proposition 6 *Let P be the feasible region of the LP relaxation of* (66) *(in which constraints* (66e) *are eliminated). Given binary x and y-values, a subproblem solution (f,z) with a fractional z- or f-value is not an extreme point of P.*

According to Proposition 6, we reformulate subproblem (66) by replacing (66e) with $z_i \ge 0$ for all $i \in \mathcal{V}$, where the upper bounds $z_i \le 1$ are unnecessary noting that no z-value exceeds 1 in any optimal solution. Let π_{ik}, $-\alpha_{0ik}$, and $-\alpha_{ijk}$ be dual variables associated with (66b), (66c), and (66d), respectively. The dual of formulation (66) is given by:

$$\eta(x,y) = \max \quad \sum_{k\in\mathcal{V}} x_k\pi_{0k} - \sum_{(i,j)\in\mathcal{A}}\sum_{k\in\mathcal{V}} y_{ij}\alpha_{ijk} \tag{67a}$$

$$\text{s.t.} \quad (\pi_{0k} - \pi_{ik}) - \alpha_{0ik} \le 0 \quad \forall i,\ k\in\mathcal{V},\ i\neq k \tag{67b}$$

$$(\pi_{ik} - \pi_{jk}) - \alpha_{ijk} \le 0 \quad \forall (i,j)\in\mathcal{A},\ k\in\mathcal{V} \tag{67c}$$

$$\sum_{k\in\mathcal{V}} \alpha_{0ik} \le 1 \quad \forall i\in\mathcal{V} \tag{67d}$$

$$\alpha_{0ik} \ge 0,\ \forall i,\ k\in\mathcal{V};\ \alpha_{ijk} \ge 0,\ \forall (i,j)\in\mathcal{A},\ k\in\mathcal{V};$$
$$\pi_{kk} = 0,\ \forall k\in\mathcal{V}, \tag{67e}$$

where (67b), (67c), and (67d) are dual constraints respectively associated with primal variables $f_{0ik},\ f_{ijk}$, and z_i. By replacing $\eta(x,y)$ with (67a) in formulation (65), we obtain a bilinear mixed-integer program with nonlinear terms of $x_k\pi_{0k}$ and $y_{ij}\alpha_{ijk}$ existing in the objective function. Observe that one optimal solution to (67) is obtained by letting $\pi_{ik} = z_k, \forall k \in \mathcal{V},\ i \in \{0\}\cup\mathcal{V},\ i \neq k$,

$\alpha_{0kk} = z_k, \forall k \in \mathcal{V}$, and all other π- and α-variables equal to zero. (This solution is feasible to (67) and has the same objective function value as that to the primal, and hence must be optimal.) It is therefore permissible to restrict our attention to solutions in which all π- and α-values belong to the interval [0,1].

Because all x- and y-variables are binary-valued, by letting $\beta_{0k} \equiv x_k \pi_{0k}$ and $\gamma_{ijk} \equiv y_{ij}\alpha_{ijk}$, we linearize these bilinear terms using the following inequalities:

$$\beta_{0k} \leq x_k,\ \beta_{0k} \leq \pi_{0k} \quad \forall k \in \mathcal{V} \tag{68a}$$

$$\gamma_{ijk} \geq y_{ij} + \alpha_{ijk} - 1,\ \gamma_{ijk} \geq 0 \quad \forall (i,j) \in \mathcal{A},\ k \in \mathcal{V}, \tag{68b}$$

where we omit constraints $\beta_{0k} \geq 0,\ \beta_{0k} \geq x_k + \pi_{0k} - 1$, $\gamma_{ijk} \leq y_{ij}$, and $\gamma_{ijk} \leq \alpha_{ijk}$ because they will not be violated by any optimal solution. The integrated MaxNum formulation is then given by

$$\begin{aligned} \textbf{MaxNum: max} \quad & \sum_{k\in\mathcal{V}} \beta_{0k} - \sum_{(i,j)\in\mathcal{A}} \sum_{k\in\mathcal{V}} \gamma_{ijk} - \frac{1}{n}\sum_{i\in\mathcal{V}} (1 - x_i) \\ \text{s.t.} \quad & \text{(65b)–(65e), (67b)–(67e), (68a)–(68b).} \end{aligned} \tag{69}$$

MinMaxC. Our next objective is to minimize the largest component size in the graph after node deletions. The MinMaxC objective function is similar to the one for MaxNum:

$$\min \left\{ \eta'(y) + \frac{1}{n}\sum_{i\in\mathcal{V}} (1 - x_i) : \text{(65b)–(65e)} \right\}, \tag{70}$$

where $\eta'(y)$ represents the largest component size given y. (Note that we can equivalently consider $\eta'(y)$ as the largest component size in G *including* all deleted nodes, because each deleted node is a singleton component. These cardinality-1 components do not increase the objective function value unless $B \geq |\mathcal{V}|$ and the problem is trivial.)

Define variables $\sigma_{ik} \in \{0,1\}$ such that $\sigma_{ik} = 1$ if nodes i and k belong to the same component, and $\sigma_{ik} = 0$ otherwise. (In particular, $\sigma_{kk} = 1,\ \forall k \in \mathcal{V}$.) Letting $\lambda = \eta'(y)$ be a variable that represents the largest component size, we give an integrated MIP formulation as:

$$\textbf{MinMaxC: min} \quad \lambda + \frac{1}{n}\sum_{i\in\mathcal{V}} (1 - x_i) \tag{71a}$$

$$\text{s.t.} \quad \text{(65b)–(65e)}$$

$$\sigma_{kk} = 1 \quad \forall k \in \mathcal{V} \tag{71b}$$

$$\sigma_{jk} - \sigma_{ik} \geq y_{ij} - 1 \quad \forall (i,j) \in \mathcal{A},\ k \in \mathcal{V} \tag{71c}$$

$$\lambda \geq \sum_{i\in\mathcal{V}} \sigma_{ik} \quad \forall k \in \mathcal{V} \tag{71d}$$

$$\sigma_{ik} \geq 0 \quad \forall i,\ k \in \mathcal{V}. \tag{71e}$$

Constraints (71d) indicate that $\lambda \geq \max_{k\in\mathcal{V}} \left\{\sum_{i\in\mathcal{V}} \sigma_{ik}\right\}$, where $\sum_{i\in\mathcal{V}} \sigma_{ik}$ calculates the cardinality of the component to which node k belongs, $\forall k \in \mathcal{V}$. At optimality, λ takes on its minimum feasible value, and thus equals $\max_{k\in\mathcal{V}} \left\{\sum_{i\in\mathcal{V}} \sigma_{ik}\right\}$. Next, we establish the following proposition to show that other constraints in (71) guarantee feasible and binary σ-values as defined.

Proposition 7 *Given binary y-values, there exists an optimal solution to* (71) *in which $\sigma_{ik} = 1$ if and only if node i and node k belong to the same component, and $\sigma_{ik} = 0$ otherwise.*

MaxMinLR. The last connectivity metric that we consider regards the minimum link construction cost for reconnecting a network. Our motivation for considering this problem is that it represents the case in which a two-stage Stackelberg game is played between an interdicting agent and some network operator. The network operator will reconnect all surviving nodes after an attack at minimum cost, while the interdictor's goal is to maximize the minimum cost that the operator incurs. Let $\widetilde{\mathcal{E}}$ be the set of edges that can be built to reconnect the graph after edge deletions, and c_{ij} be the link construction cost associated with edge $(i,j),\ \forall (i,j) \in \widetilde{\mathcal{E}}$.

We begin by showing that MaxNum is a special case of MaxMinLR. Note that any MaxMinLR instance in which $\widetilde{\mathcal{E}} = \mathcal{E}$, and $c_{ij} = 1,\ \forall (i,j) \in \widetilde{\mathcal{E}}$, is identical to a MaxNum instance on the same graph: The number of components in an optimal MaxNum solution is one fewer than the number of edges required to reconnect the graph. (This observation is true regardless of whether we treat deleted nodes as entities that need to be reconnected in the MaxMinLR problem.) Thus, because MaxNum is a special case of MaxMinLR, and MaxNum is strongly $\mathcal{NP}$-hard, then MaxMinLR is also strongly $\mathcal{NP}$-hard. By contrast to MaxNum and MinMaxC, Theorem 15 shows that when deleted nodes do not need to be reconnected, MaxMinLR remains hard even when G is a tree.

Theorem 15 *MaxMinLR (without reconnecting deleted nodes) is strongly $\mathcal{NP}$-hard, even if c_{ij} is binary-valued for all $(i,j) \in \widetilde{\mathcal{E}}$, and if G is a tree.*

We model MaxMinLR as an MIP by creating a directed auxiliary graph $\widetilde{G}(\mathcal{V}, \mathcal{A})$, in which two directed arcs (i,j) and (j,i) appear in $\mathcal{A}$ associated with each edge $(i,j) \in \mathcal{E}$. Meanwhile, two directed arcs (i,j) and (j,i) appear in a potential arc set $\widetilde{\mathcal{A}}$ for each edge $(i,j) \in \widetilde{\mathcal{E}}$ that may be used to reconnect the graph. Recall that $\widetilde{\mathcal{V}}$ is the set of all active nodes. We formulate the problem of determining the minimum connection cost on $\widetilde{G}(\mathcal{V}, \mathcal{A})$ as a multi-commodity network design problem, which routes $(|\widetilde{\mathcal{V}}| - 1)$ paths from some active node q to all other active nodes (one path each from node q to node i, $\forall i \in \widetilde{\mathcal{V}} \setminus \{q\}$). Without loss of generality, we specify source node q as the smallest-indexed active node. Define binary variables w_i, such that $w_i = 1$ if node i is the smallest-indexed node in $\widetilde{\mathcal{V}}$, and $w_i = 0$ otherwise. In the master problem, we establish the definition of w by using the following inequalities:

$$w_i \leq x_i \quad \forall i \in \mathcal{V} \tag{72a}$$

$$w_i \leq 1 - x_k \quad \forall i,\ k \in \mathcal{V} :\ k < i \tag{72b}$$

$$\sum_{i \in \mathcal{V}} w_i = 1, \tag{72c}$$

where constraints (72a) ensure $w_i = 0$ if node i has been deleted, and constraints (72b) enforce $w_i = 0$ if any node k having a smaller index than i (i.e., $k < i$) has not been deleted. Constraint (72c) forces one eligible w_i to take a value of 1. The master problem of MaxMinLR is similar to (65) except that the objective maximizes $\eta''(w,x,y) - 1/n \sum_{i \in \mathcal{V}} (1 - x_i)$, where $\eta''(w,x,y)$

is the minimum link construction cost given binary values w, x, and y, and the constraint set includes (72a)–(72c).

Define $FS'(i) = \{j : (i,j) \in \mathcal{A} \cup \widetilde{\mathcal{A}}\}$ as the set of nodes potentially adjacent from node i, and $RS'(i) = \{j : (j,i) \in \mathcal{A} \cup \widetilde{\mathcal{A}}\}$ as the set of nodes potentially adjacent to i, $\forall i \in \mathcal{V}$. Let f_{ijk} be the flow on arc $(i,j) \in \mathcal{A} \cup \widetilde{\mathcal{A}}$ corresponding to paths from node q to node k, $\forall k \in \mathcal{V}$. Define binary variables $z_{ij} \in \{0,1\}$, such that $z_{ij} = 1$ if we construct arc (i,j), and $z_{ij} = 0$ otherwise, $\forall (i,j) \in \widetilde{\mathcal{A}}$. The MaxMinLR subproblem is given by

$$\eta''(w,x,y) = \min \quad \sum_{(i,j)\in\widetilde{\mathcal{A}}} c_{ij} z_{ij} \tag{73a}$$

$$\text{subject to:} \quad \sum_{j\in FS'(i)} f_{ijk} - \sum_{j\in RS'(i)} f_{jik} \geq w_i - (1 - x_k) \quad \forall i,\ k \in \mathcal{V},\ i < k \tag{73b}$$

$$f_{ijk} \leq z_{ij} + y_{ij} \quad \forall (i,j) \in \mathcal{A} \cup \widetilde{\mathcal{A}},\ k \in \mathcal{V} \tag{73c}$$

$$z_{ij} \leq y_{ij} \quad \forall (i,j) \in \widetilde{\mathcal{A}} \tag{73d}$$

$$z_{ij} \in \{0,1\} \quad \forall (i,j) \in \widetilde{\mathcal{A}} \tag{73e}$$

$$f_{ijk} \geq 0 \quad \forall (i,j) \in \mathcal{A} \cup \widetilde{\mathcal{A}},\ k \in \mathcal{V}, \tag{73f}$$

where constraints (73b) represent multi-commodity flow balance conditions at each node $i \in \widetilde{\mathcal{V}}$, which require the existence of a path from node q to node k for all active $k \in \widetilde{\mathcal{V}}$. Note that these constraints are only stated for all $i,\ k \in \mathcal{V}, i < k$, because if $w_i = 1$, then $x_1 = \cdots = x_{i-1} = 0$ due to (72b). Constraints (73c) force $f_{ijk} = 0$ if $(i,j) \in \mathcal{A} \cup \widetilde{\mathcal{A}}$ is neither connected (i.e., $y_{ij} = 0$) nor constructed (i.e., $z_{ij} = 0$), where $y_{ij} \equiv 0$ if $(i,j) \notin \mathcal{A}$ and $z_{ij} \equiv 0$ if $(i,j) \notin \widetilde{\mathcal{A}}$. Constraints (73d) stipulate that no arc incident to any deleted node is constructed. Constraints (73e) and (73f) state logical conditions on the variables.

Proposition 8 *Let P' be the feasible region of the LP relaxation of* (73) *in which integrality constraints* (73e) *are eliminated. Given binary values of w and y, a subproblem solution (f,z) with fractional f or z is not an extreme point of P'.*

The proof of Proposition 8 is similar to the proof of Proposition 6 and is hence omitted. Therefore, to formulate MaxMinLR as a single MIP, we replace $\eta''(w,x,y)$ in the master problem by the dual of the LP relaxation of (73), and rewrite all bilinear terms as a series of linear inequalities similar to (68a) and (68b).

Let τ_{ik}, $-\theta_{ijk}$, and $-\mu_{ij}$ be dual variables associated with (73b), (73c), and (73d), respectively. In particular, define $\tau_{ik} \equiv 0$ for all $i,\ k \in \mathcal{V},\ i \geq k$. We present the integrated MaxMinLR MIP model as follows.

$$\max \quad \sum_{i\in\mathcal{V}} \sum_{k\in\mathcal{V},k>i} \left(\alpha'_{ik} + \beta'_{ik} - \tau_{ik}\right) - \sum_{(i,j)\in\mathcal{A}\cup\widetilde{\mathcal{A}}} \sum_{k\in\mathcal{V}} \gamma'_{ijk} - \sum_{(i,j)\in\widetilde{\mathcal{A}}} \delta'_{ij} - \frac{1}{n} \sum_{i\in\mathcal{V}} (1 - x_i) \tag{74a}$$

$$\text{s.t.} \quad \text{(65b)–(65e), (72a)–(72c)}$$

$$\tau_{jk} - \tau_{ik} - \theta_{ijk} \leq 0 \quad \forall (i,j) \in \mathcal{A} \cup \widetilde{\mathcal{A}},\ k \in \mathcal{V} \tag{74b}$$

$$\sum_{k\in\mathcal{V}} \theta_{ijk} - \mu_{ij} \leq c_{ij} \quad \forall (i,j) \in \widetilde{\mathcal{A}} \tag{74c}$$

$$\alpha'_{ik} \leq w_i,\ \alpha'_{ik} \leq \tau_{ik} \quad \forall i,\ k \in \mathcal{V},\ i < k \tag{74d}$$

$$\beta'_{ik} \leq x_k,\ \beta'_{ik} \leq \tau_{ik} \quad \forall i,\ k \in \mathcal{V},\ i < k \tag{74e}$$

$$\gamma'_{ijk} \geq y_{ij} + \theta_{ijk} - 1,\ \gamma'_{ijk} \geq 0 \quad \forall (i,j) \in \mathcal{A} \cup \widetilde{\mathcal{A}},\ k \in \mathcal{V} \tag{74f}$$

$$\delta'_{ij} \geq y_{ij} + \mu_{ij} - 1,\ \delta'_{ij} \geq 0 \quad \forall (i,j) \in \widetilde{\mathcal{A}} \tag{74g}$$

$$\tau_{ik} \geq 0 \quad \forall i,\ k \in \mathcal{V},\ i < k; \quad \theta_{ijk} \geq 0 \quad \forall (i,j) \in \mathcal{A} \cup \widetilde{\mathcal{A}},\ k \in \mathcal{V};$$
$$\mu_{ij} \geq 0 \quad \forall (i,j) \in \widetilde{\mathcal{A}}. \tag{74h}$$

Constraints (74b) and (74c) are dual constraints associated with primal variables $f_{ijk},\ \forall (i,j) \in \mathcal{A} \cup \widetilde{\mathcal{A}},\ k \in \mathcal{V}$ and $z_{ij},\ \forall (i,j) \in \widetilde{\mathcal{A}}$, respectively. We define $\alpha'_{ik} \equiv w_i \tau_{ik}$, $\beta'_{ik} \equiv x_k \tau_{ik}$, $\gamma'_{ijk} \equiv y_{ij}\theta_{ijk}$, and $\delta'_{ij} \equiv y_{ij}\mu_{ij}$ in the objective, and enforce these relationships via constraints (74d), (74e), (74f), and (74g), respectively, as before.

Bounds and Inequalities for MaxNum and MinMaxC. We now discuss strategies for bounding the optimal objective function values for MaxNum and MinMaxC, and using the obtained bounds to tighten the MIP formulations for these problems. Shen and Smith (2012, Networks) prescribe polynomial-time optimal DP algorithms for MaxNum and MinMaxC on k-hole-graphs (where k is a constant). We do not carry out this analysis for MaxMinLR, as Theorem 15 indicates that it is not solvable (in general) in polynomial time, even when G is a tree, unless $\mathcal{P} = \mathcal{NP}$.

We begin by describing mechanisms for bounding the connectivity objectives, i.e., the number of components in MaxNum, and the largest component size in MinMaxC. Denote x_G and x'_G as optimal solutions to MaxNum and MinMaxC on G, respectively. Also, let $\eta_G(x)$ equal the number of components, and $\eta'_G(x)$ equal the largest component size, after we delete a subset of nodes (and their incident edges) associated with solution x on graph G. We first establish the following proposition.

Lemma 11 *For any subgraph $G_S(\mathcal{V}, \mathcal{E}_S)$ of G, we have:*

$$\eta_G(x_{G_S}) \leq \eta_G(x_G) \leq \eta_{G_S}(x_{G_S}) \tag{75a}$$

$$\eta'_G(x'_{G_S}) \geq \eta'_G(x'_G) \geq \eta'_{G_S}(x'_{G_S}). \tag{75b}$$

In our implementation, we specify *a priori* some value $k_{\max}$ that denotes the maximum number of holes that we allow in any induced subgraph of G, and generate a series of m subgraphs of G, denoted as $H^1, \ldots, H^m$, in which H^l is a k_l-hole-graph ($k_l \leq k_{\max}$), $\forall l = 1, \ldots, m$. According to Lemma 11, the following bounds are valid:

$$\max_{l=1,\ldots,m} \left\{\eta_G\left(x_{H^l}\right)\right\} \leq \eta_G(x_G) \leq \min_{l=1,\ldots,m} \left\{\eta_{H^l}\left(x_{H^l}\right)\right\} \tag{76a}$$

$$\min_{l=1,\ldots,m} \left\{\eta'_G\left(x'_{H^l}\right)\right\} \geq \eta'_G(x'_G) \geq \max_{l=1,\ldots,m} \left\{\eta'_{H^l}\left(x'_{H^l}\right)\right\}. \tag{76b}$$

Given the bounds established in (76a) and (76b), we now turn our attention to employing these bounds within a graph partitioning strategy that generates valid inequalities for MaxNum and MinMaxC. Given $G(\mathcal{V}, \mathcal{E})$, we partition $\mathcal{V}$ into m nonempty subsets $V_1, \ldots, V_m$, such that $V_i \cap V_j = \emptyset$ for all $i,\ j = 1, \ldots, m,\ i \neq j$ and $\bigcup_{i=1}^{m} V_i = \mathcal{V}$. Each partition V_i yields a subgraph $G_i(V_i, E_i)$, in which $E_i = \{(u, v) \in \mathcal{E} \mid u, v \in V_i\}$ is induced by nodes in V_i. (Note that if G is connected and $m \geq 2$, then $\bigcup_{i=1}^{m} E_i \subset \mathcal{E}$.) Let k_i be the number of holes in each subgraph $G_i,\ \forall i = 1, \ldots, m$.

We then execute the DP algorithm of Shen and Smith on each k_i-hole subgraph G_i given a node deletion budget of B. As is typical in DP schemes, we obtain the optimal connectivity objective values corresponding to every node deletion budget value $B_i = 0, \ldots, B$. These values allow us to construct functions that reflect relationships between the connectivity objective values and budgets in MaxNum and MinMaxC over G_i, which yield valid inequalities for their respective MIPs.

We first present the development of our valid inequalities in the context of the MaxNum problem. Let $\eta_i(B_i)$ be the maximum number of components that we can obtain in G_i given deletion budget $B_i = 0, \ldots, B$. Let variable η represent the optimal connectivity objective value for solving MaxNum on G, and let η_i be a variable representing the optimal connectivity objective value for solving MaxNum on $G_i,\ \forall i = 1, \ldots, m$. We construct a piecewise-linear concave envelope function $g_i(B_i)$ of $\eta_i(B_i)$, such that $\eta_i(B_i) \leq g_i(B_i)$ for each $B_i = 0, \ldots, B$. We use the tightest such function possible, in which every linear segment of $g_i(B_i)$ touches at least two points on the function $\eta_i(B_i)$, assuming that $B \geq 1$. We append the following system of valid inequalities into the MaxNum formulation, where B_i is now released as a variable representing the number of node deletions that take place over V_i.

$$\eta - \sum_{i=1}^{m} \eta_i \leq 0 \tag{77a}$$

$$\eta_i - g_i(B_i) \leq 0 \quad \forall i = 1, \ldots, m \tag{77b}$$

$$B_i = \sum_{j \in V_i} (1 - x_j) \quad \forall i = 1, \ldots, m, \tag{77c}$$

where (77a) is due to the fact that the partition procedure automatically eliminates all edges between each pair of $G_i,\ \forall i = 1, \ldots, m$, and thus it creates at least as many components as solving the original problem on G. Constraints (77b) are nonlinear constraints that can be substituted by a set of linear functions, each of which corresponds to a segment of $g_i(B_i)$, for all $i = 1, \ldots, m$. Constraints (77c) define B_i as the number of node deletions taking place in set V_i.

8 Defending and Interdicting NP-hard Problems

Lozano and Smith (2015) consider defender-attacker-defender problems that are modeled as three-level, two-player Stackelberg games. In the first stage a defender (also known as the "owner" or "operator") can fortify a subset of assets, while in the second stage an attacker (often called the "interdictor") destroys a subset of the unprotected assets. The attacker's goal in the second stage is to maximize damage to the defender's objective, which is determined by solving an optimization problem in the third stage, using the surviving assets from the initial system.

Formally, let $\mathbf{w}$, $\mathbf{x}$, and $\mathbf{y}$ be the decision variables for the first-, second-, and third-stage problems, respectively. We assume that the third-stage problem can take any general form, while the first- and second-stage problems include only binary variables, i.e., $\mathbf{w} \in \{0,1\}^{n_\mathbf{w}}$ and $\mathbf{x} \in \{0,1\}^{n_\mathbf{x}}$, where $n_\mathbf{w}$ ($n_\mathbf{x}$) is the number of variables required to model asset fortification (attack). Let $\mathcal{W}$ be the set of feasible solutions to the first-stage problem. Let $\mathcal{X}(\mathbf{w})$ be the set of feasible second-stage solutions given a defense vector $\mathbf{w}$, and let $\mathcal{Y}(\mathbf{x})$ be the set of feasible third-stage solutions for a given attack vector $\mathbf{x}$. Also, define $\mathcal{X} = \bigcup_{\mathbf{w}\in\mathcal{W}} \mathcal{X}(\mathbf{w})$ and $\mathcal{Y} = \bigcup_{\mathbf{x}\in\mathcal{X}} \mathcal{Y}(\mathbf{x})$, i.e., $\mathcal{X}$ and $\mathcal{Y}$ are the set of all possible second- and third-stage feasible solutions, respectively. Finally, let $f(\mathbf{y})$ be the defender's objective function. We study problems of the form:

$$\mathcal{P}: \quad z^* = \min_{\mathbf{w}\in\mathcal{W}} \max_{\mathbf{x}\in\mathcal{X}(\mathbf{w})} \min_{\mathbf{y}\in\mathcal{Y}(\mathbf{x})} f(\mathbf{y}), \tag{78}$$

We refer to the first-, second-, and third-stage problems as *fortification*, *attack*, and *recourse* problems, respectively.

These problems have multiple applications in areas such as military and homeland security operations, facility protection, survivable network design, and power grid protection. At a more abstract level, interdiction problems can often be modeled as games that take place over networks having well-studied recourse problems. Some of these network interdiction problems include shortest path, maximum flow, and multicommodity flow studies. Of particular interest are previous studies on defender-attacker-defender problems that fit within our problem framework.

We present a novel backward sampling framework for solving three- (and two-) stage interdiction problems in which the recourse problem can take any form (e.g., it can be nonlinear, and can have integer variables), provided that all variables in the first two stages are restricted to be binary-valued. This framework is primarily designed to improve the solution of the interdiction problem, by solving relatively easy interdiction problem relaxations in which the defender is restricted to choose its recourse actions from a sample of the third-stage solution space. These problems provide upper bounds on the optimal interdiction solution; lower bounds can then be obtained by fixing an interdiction solution and optimizing the (original) recourse problem as a function of the fixed interdiction actions. This framework avoids linearizing a (potentially large) bilinear program, and also eliminates the need for applying combinatorial Benders' cuts at the interdiction stage (although we still require them to solve the fortification problem).

Using our framework, we construct an algorithm for the shortest path interdiction problem with fortification (SPIF) that compares favorably to the current state-of-the-art algorithm, finding optimal solutions over random grid networks having up to 3,600 nodes and 17,000 arcs, and over real-road networks having up to 300,000 nodes and more than 1,000,000 arcs. We also consider the capacitated lot sizing interdiction problem with fortification (CLSIF), in which the NP-hard third-stage problem is modeled as a MIP. We extend our framework to solve the CLSIF, and demonstrate its ability to solve instances of this new problem class.

The Backward Sampling Framework. The core idea behind the backward sampling framework is to iteratively sample the third-stage solution space so that instead of solving the original

problem $\mathcal{P}$ directly, we solve *restricted problems* defined over smaller recourse solution spaces. The sampling procedure selects a subset of third-stage solutions $\hat{\mathcal{Y}} \subseteq \mathcal{Y}$, and throughout the algorithm augments $\hat{\mathcal{Y}}$ with new third-stage solutions from $\mathcal{Y}$. The sampling procedure would ideally be able to quickly identify several near-optimal solutions; however, we do not require this procedure to guarantee the generation of any new solutions in order for our framework to converge to an optimal solution. An appropriate strategy would tailor the sampling procedure for the problem at hand, as would be done for heuristic approaches.

For any attack vector $\mathbf{x}$ and third-stage solution sample $\hat{\mathcal{Y}}$, we denote by $\hat{\mathcal{Y}}(\mathbf{x}) \equiv \hat{\mathcal{Y}} \cap \mathcal{Y}(\mathbf{x})$ the subset of solutions that belong to $\hat{\mathcal{Y}}$ and are feasible given the attack vector $\mathbf{x}$. Anticipating the case for which there exists an attack $\mathbf{x} \in \mathcal{X}$ for which $\hat{\mathcal{Y}}(\mathbf{x}) = \emptyset$, we seed $\hat{\mathcal{Y}}$ with an artificial third-stage solution $\mathbf{y}^a$ that cannot be interdicted and has objective value $f(\mathbf{y}^a) = \infty$. This artificial solution ensures that $\hat{\mathcal{Y}}(\mathbf{x}) \neq \emptyset$ for any $\mathbf{x} \in \mathcal{X}$.

Consider any feasible defense vector $\mathbf{w} \in \mathcal{W}$ and let

$$\mathcal{Q}(\mathbf{w}) : z^I(\mathbf{w}) = \max_{\mathbf{x} \in \mathcal{X}(\mathbf{w})} \min_{\mathbf{y} \in \mathcal{Y}(\mathbf{x})} f(\mathbf{y}) \tag{79}$$

be its associated two-level interdiction problem. Note that if there exists a defense $\mathbf{w} \in \mathcal{W}$ such that $\mathcal{X}(\mathbf{w}) = \emptyset$, then problem (79) is not defined. Hence, without loss of generality, we assume that $\mathcal{X}(\mathbf{w}) \neq \emptyset,\ \forall \mathbf{w} \in \mathcal{W}$.

Let $\hat{\mathcal{Y}} \subseteq \mathcal{Y}$ be any third-stage solution sample and

$$\mathcal{Q}(\mathbf{w}, \hat{\mathcal{Y}}) : z^I(\mathbf{w}, \hat{\mathcal{Y}}) = \max_{\mathbf{x} \in \mathcal{X}(\mathbf{w})} \min_{\mathbf{y} \in \hat{\mathcal{Y}}(\mathbf{x})} f(\mathbf{y}) \tag{80}$$

be the restricted problem in which recourse (third-stage) decisions are restricted to $\hat{\mathcal{Y}}$. The following result establishes that solving a restricted problem $\mathcal{Q}(\mathbf{w}, \hat{\mathcal{Y}})$ yields a valid upper bound on $z^I(\mathbf{w})$, which is in turn a valid upper bound on z^*.

Proposition 9 *Consider any* $\mathbf{w} \in \mathcal{W}$ *and third-stage solution sample* $\hat{\mathcal{Y}} \subseteq \mathcal{Y}$. *Then we have* $z^I(\mathbf{w}, \hat{\mathcal{Y}}) \geq z^I(\mathbf{w}) \geq z^*$.

All proofs can be found within the work of Lozano and Smith (2015).

We now establish conditions under which we can obtain an optimal solution to $\mathcal{Q}(\mathbf{w})$, for some $\mathbf{w} \in \mathcal{W}$, from a restricted problem $\mathcal{Q}(\mathbf{w}, \hat{\mathcal{Y}})$. First, let $(\hat{\mathbf{x}}, \hat{\mathbf{y}})$ be an optimal solution to the restricted problem $\mathcal{Q}(\mathbf{w}, \hat{\mathcal{Y}})$. We say that $z^I(\mathbf{w}, \hat{\mathcal{Y}})$ is the *perceived damage* of $\hat{\mathbf{x}}$ given $\hat{\mathcal{Y}}$, because the interdictor perceives that the recourse decision must come from $\hat{\mathcal{Y}}$. However, the defender may instead select from uninterdicted solutions in $\mathcal{Y}$, and so we define the *real damage* of attack $\hat{\mathbf{x}}$ over the original third-stage solution space $\mathcal{Y}$ as

$$z^R(\hat{\mathbf{x}}) = \min_{\mathbf{y} \in \mathcal{Y}(\hat{\mathbf{x}})} f(\mathbf{y}). \tag{81}$$

Observe that $z^R(\hat{\mathbf{x}}) \leq z^I(\mathbf{w}) \leq z^I(\mathbf{w}, \hat{\mathcal{Y}})$ for any $\hat{\mathbf{x}} \in \mathcal{X}(\mathbf{w})$. Proposition 10 states a condition in which an optimal solution to $\mathcal{Q}(\mathbf{w}, \hat{\mathcal{Y}})$ also optimizes $\mathcal{Q}(\mathbf{w})$.

Proposition 10 *Let* $\mathbf{w} \in \mathcal{W}$ *be a feasible defense,* $\hat{\mathcal{Y}}$ *be a third-stage solution sample, and* $(\hat{\mathbf{x}}, \hat{\mathbf{y}})$ *be an optimal solution to* $\mathcal{Q}(\mathbf{w}, \hat{\mathcal{Y}})$*. If* $z^I(\mathbf{w}, \hat{\mathcal{Y}}) = z^R(\hat{\mathbf{x}})$*, then* $(\hat{\mathbf{x}}, \hat{\mathbf{y}})$ *optimizes* $\mathcal{Q}(\mathbf{w})$*.*

Our algorithm uses these results to solve $\mathcal{Q}(\mathbf{w})$, given $\mathbf{w} \in \mathcal{W}$, by iteratively solving restricted problems $\mathcal{Q}(\mathbf{w}, \hat{\mathcal{Y}}^i)$ defined over different third-stage samples $\hat{\mathcal{Y}}^i \subseteq \mathcal{Y}$. Algorithm 15 presents this approach, in which each iteration i yields an upper bound UB_i on $z^I(\mathbf{w})$ from solving $\mathcal{Q}(\mathbf{w}, \hat{\mathcal{Y}}^i)$, and a lower bound LB_i on $z^I(\mathbf{w})$ by obtaining $z^R(\hat{\mathbf{x}})$, for some $\hat{\mathbf{x}} \in \mathcal{X}(\mathbf{w})$. As we will demonstrate, the sequence of UB_i-values is nonincreasing, although the LB_i-values need not be monotone. The main while-loop (line 2) is executed until the optimality condition described in Proposition 10 is met. Line 4 solves the restricted problem $\mathcal{Q}(\mathbf{w}, \hat{\mathcal{Y}}^i)$ defined over the current sample $\hat{\mathcal{Y}}^i$. Line 5 calculates the real damage $z^R(\hat{\mathbf{x}})$ for attack $\hat{\mathbf{x}}$ and sets LB_i equal to this value (see Remark 1 for additional explanation). Line 6 creates $\hat{\mathcal{Y}}^{i+1}$ by including solutions in $\hat{\mathcal{Y}}^i$ along with $\hat{\mathbf{y}}^*$, i.e., an optimal third-stage response to attack $\hat{\mathbf{x}}$.

If the perceived damage obtained is less than the upper bound at the previous iteration, then a new upper bound on $z^I(\mathbf{w})$ has been obtained, and the algorithm executes lines 8–9. Line 8 removes from $\hat{\mathcal{Y}}^{i+1}$ solutions whose objective value is greater than UB_i, and line 9 attempts to add new solutions to $\hat{\mathcal{Y}}^{i+1}$ from $\mathcal{Y}_{UB_i} \equiv \{\mathbf{y} \in \mathcal{Y} \mid f(\mathbf{y}) \leq UB_i\}$. If the optimality condition in line 11 is satisfied, then line 12 returns an optimal solution.

Algorithm 15: Solving bilevel interdiction problem $Q(\mathbf{w})$ via backward sampling

Input : Problem $\mathcal{P}$ and a feasible defense $\mathbf{w} \in \mathcal{W}$
Output: An optimal solution to $\mathcal{Q}(\mathbf{w})$

1 Initialize $UB_0 = \infty$ and $LB_0 = -\infty$, select $\hat{\mathcal{Y}}^1 \subseteq \mathcal{Y}$ as a sampling of the third-stage solution space, and set counter $i = 0$;
2 **while** $\text{LB}_i < \text{UB}_i$ **do**
3 Set $i = i + 1$;
4 Solve $UB_i = z^I(\mathbf{w}, \hat{\mathcal{Y}}^i) = \max_{\mathbf{x} \in \mathcal{X}(\mathbf{w})} \min_{\mathbf{y} \in \hat{\mathcal{Y}}^i(\mathbf{x})} f(\mathbf{y})$ and obtain an optimal solution $(\hat{\mathbf{x}}, \hat{\mathbf{y}})$;
5 Solve $LB_i = z^R(\hat{\mathbf{x}}) = \min_{\mathbf{y} \in \mathcal{Y}(\hat{\mathbf{x}})} f(\mathbf{y})$ and obtain an optimal solution $\hat{\mathbf{y}}^*$;
6 Set $\hat{\mathcal{Y}}^{i+1} = \hat{\mathcal{Y}}^i \cup \{\hat{\mathbf{y}}^*\}$;
7 **if** $\text{UB}_i < \text{UB}_{i-1}$ **then**
8 Remove from $\hat{\mathcal{Y}}^{i+1}$ all solutions having objective value greater than UB_i;
9 Add to $\hat{\mathcal{Y}}^{i+1}$ a subset of new solutions in $\mathcal{Y}_{UB_i}$;
10 **end**
11 **if** $\text{LB}_i = \text{UB}_i$ **then**
12 Terminate with solution $(\hat{\mathbf{x}}, \hat{\mathbf{y}})$;
13 **end**
14 **end**

Proposition 11 shows that the sequence of UB_i-values obtained is nonincreasing, and Proposition 12 states the finiteness and correctness of the proposed algorithm.

Proposition 11 *The upper bounds UB_i produced by Algorithm 15 are nonincreasing, and at iteration i, $\hat{\mathcal{Y}}^1_{UB_i} \subseteq \hat{\mathcal{Y}}^2_{UB_i} \subseteq \cdots \subseteq \hat{\mathcal{Y}}^{i+1}_{UB_i}$, where $\hat{\mathcal{Y}}^j_{UB_i} \equiv \{\mathbf{y} \in \hat{\mathcal{Y}}^j \mid f(\mathbf{y}) \leq UB_i\}$.*

Proposition 12 *Algorithm 15 terminates finitely with an optimal solution.*

Optimizing the Defense Decisions. We now propose an approach to solve the three-level problem $\mathcal{P}$. This approach is based on the identification of *critical attacks*, i.e., attacks that must be blocked in order to improve the defender's incumbent objective value. Formally, we define a critical attack as any attack $\hat{\mathbf{x}}$ such that its real damage $z^R(\hat{\mathbf{x}})$ is greater than or equal to a target upper bound $\bar{z}$. Our approach adds a *covering constraint* $\mathbf{w}^\intercal \hat{\mathbf{x}} \geq 1$ to the fortification problem for each critical attack $\hat{\mathbf{x}}$, which states that at least one of the assets attacked by $\hat{\mathbf{x}}$ must be fortified.

Proposition 13 *For problem $\mathcal{P}$ having optimal objective value z^*, consider any attack $\hat{\mathbf{x}} \in \mathcal{X}$. If $z^* < z^R(\hat{\mathbf{x}})$, then any optimal solution $(\mathbf{w}^*, \mathbf{x}^*, \mathbf{y}^*)$ satisfies $\mathbf{w}^{*\intercal} \hat{\mathbf{x}} \geq 1$.*

These covering constraints can be seen as a general case of the *combinatorial Benders' cut* (Codato and Fischetti, 2006) where the fortification problem acts as a master problem and $\mathcal{Q}(\hat{\mathbf{w}})$ as a subproblem. Similar so-called *supervalid inequalies* were introduced by Israeli and Wood (2002) for a two-level shortest path interdiction problem.

Our approach explores different defense vectors $\hat{\mathbf{w}} \in \mathcal{W}$ and solves the associated interdiction problems $\mathcal{Q}(\hat{\mathbf{w}})$ with a variation of Algorithm 15 that stops whenever it identifies a critical attack. When such an attack is identified, the algorithm adds a covering constraint to the fortification problem, forcing the defender to block each identified critical attack. When the fortification problem becomes infeasible, the algorithm terminates with the incumbent solution being optimal. This process must eventually terminate with an infeasible first-stage problem because $\mathcal{X}(\mathbf{w}) \neq \emptyset,\ \forall \mathbf{w} \in \mathcal{W}$, by assumption.

Algorithm 16 presents the proposed approach. Let $\mathcal{C}$ be the set of covering constraints added to the fortification problem and $\mathcal{W}(\mathcal{C}) \equiv \{\mathbf{w} \in \mathcal{W} \mid \mathbf{w} \text{ satisfies all constraints in } \mathcal{C}\}$. The algorithm starts with $\mathcal{C} = \emptyset$ and a global upper bound $\bar{z} = \infty$. The main while-loop (in line 2) is executed until the fortification problem becomes infeasible. The two main steps inside this while-loop are selecting a feasible defense $\hat{\mathbf{w}} \in \mathcal{W}(\mathcal{C})$ (in line 4), and solving its associated interdiction problem $\mathcal{Q}(\hat{\mathbf{w}})$ with a variation of Algorithm 15 (lines 5–22). The inner while-loop (in line 6) is executed until $LB_i = z^R(\hat{\mathbf{x}}) \geq \bar{z}$, for some $\hat{\mathbf{x}} \in \mathcal{X}(\hat{\mathbf{w}})$, indicating that $\hat{\mathbf{x}}$ is a critical attack. At this point, Algorithm 16 stops solving $\mathcal{Q}(\hat{\mathbf{w}})$ and adds a covering constraint to $\mathcal{C}$. Finally, lines 7–21 replicate Algorithm 15, except for updating the global upper bound $\bar{z}$ (in line 12), adding a covering constraint to $\mathcal{C}$ if a critical attack is identified (in lines 16–18), and updating the incumbent solution when an optimal solution to $\mathcal{Q}(\hat{\mathbf{w}})$ has an objective value equal to $\bar{z}$ (in lines 19–21).

Algorithm 16 terminates finitely because each critical attack $\hat{\mathbf{x}} \in \mathcal{X}$ triggers the generation of a covering constraint to $\mathcal{C}$, which excludes the fortification action $\hat{\mathbf{w}}$ from $\mathcal{W}(\mathcal{C})$. Finite termination of the algorithm then follows from the finiteness of $\mathcal{W}$ and from Proposition 12.

Algorithm 16: Backward sampling framework

Input : Problem $\mathcal{P}$
Output: An optimal solution to $\mathcal{P}$

Initialize the global upper bound $\bar{z} = \infty$ and covering constraints set $\mathcal{C} = \emptyset$;
Select $\hat{\mathcal{Y}}^1 \subseteq \mathcal{Y}$ as a sampling of the third-stage solution space and set counter $i = 0$;
while $\mathcal{W}(\mathcal{C}) \neq \emptyset$ **do**
    Select $\hat{\mathbf{w}} \in \mathcal{W}(\mathcal{C})$;
    Initialize $UB_i = \infty$ and $LB_i = -\infty$;
    while $\text{LB}_i < \bar{z}$ **do**
        Set $i = i + 1$;
        Solve $UB_i = z^I(\mathbf{w}, \hat{\mathcal{Y}}^i) = \max_{\mathbf{x} \in \mathcal{X}(\mathbf{w})} \min_{\mathbf{y} \in \hat{\mathcal{Y}}^i(\mathbf{x})} f(\mathbf{y})$ and obtain an optimal solution $(\hat{\mathbf{x}}, \hat{\mathbf{y}})$;
        Solve $LB_i = z^R(\hat{\mathbf{x}}) = \min_{\mathbf{y} \in \mathcal{Y}(\hat{\mathbf{x}})} f(\mathbf{y})$ and obtain an optimal solution $\hat{\mathbf{y}}^*$;
        Set $\hat{\mathcal{Y}}^{i+1} = \hat{\mathcal{Y}}^i \cup \{\hat{\mathbf{y}}^*\}$;
        if $\text{UB}_i < \bar{z}$ **then**
            Update global upper bound $\bar{z} \leftarrow UB_i$;
            Remove from $\hat{\mathcal{Y}}^{i+1}$ all solutions having objective value greater than UB_i;
            Add to $\hat{\mathcal{Y}}^{i+1}$ a subset of new solutions in $\mathcal{Y}_{UB_i}$;
        end
        if $\text{LB}_i \geq \bar{z}$ **then**
            Add the covering constraint $\mathbf{w}^\intercal \hat{\mathbf{x}} \geq 1$ to $\mathcal{C}$;
        end
        if $\text{LB}_i = \text{UB}_i = \bar{z}$ **then**
            Update the incumbent solution $(\bar{\mathbf{w}}, \bar{\mathbf{x}}, \bar{\mathbf{y}}) \leftarrow (\hat{\mathbf{w}}, \hat{\mathbf{x}}, \hat{\mathbf{y}})$;
        end
    end
end
Return $(\bar{\mathbf{w}}, \bar{\mathbf{x}}, \bar{\mathbf{y}})$;

The correctness of Algorithm 16 results directly from Propositions 9 and 13. Note that the upper bound $\bar{z}$ is nonincreasing throughout the execution of the algorithm. Proposition 13 states that each of the covering constraints is necessary in order to achieve an objective value less than $\bar{z}$. As a result, once $\mathcal{W}(\mathcal{C})$ becomes empty we conclude that $z^* \geq \bar{z}$. Since $\bar{z}$ is an upper bound, we also have that $z^* \leq \bar{z}$, which guarantees that the algorithm terminates with the optimal value $\bar{z} = z^*$. For any $\hat{\mathbf{w}}$ that reduces $\bar{z}$, the algorithm solves $\mathcal{Q}(\hat{\mathbf{w}})$ to optimality, i.e., until $LB_i = UB_i = \bar{z}$, and updates the incumbent solution. As a result, the algorithm terminates with an optimal incumbent solution since its objective value is equal to $\bar{z} = z^*$.

Accelerating the Algorithm. We now describe a mechanism designed to reduce the number of restricted interdiction problems that are solved to optimality. The idea is to "pause" the exploration of any $\hat{\mathbf{w}} \in \mathcal{W}$ whenever the potential relative improvement to the current global upper bound is sufficiently small. At this point, we add a tentative covering constraint to the fortification problem,

guessing that the best known attack $\hat{\mathbf{x}}$ corresponding to $\hat{\mathbf{w}}$ is critical (which it will indeed be if the global upper bound is reduced by a relatively small amount). We store $\hat{\mathbf{w}}$ in a waiting list to be revisited later in the execution of the algorithm, at which time we either confirm that $\hat{\mathbf{x}}$ was critical and discard $\hat{\mathbf{w}}$ from the waiting list, or conclude that the attack may not be critical and continue exploring $\hat{\mathbf{w}}$.

Formally, let $\mathcal{C}$ be the set of covering constraints derived from (known) critical attacks and $\mathcal{C}^\psi$ be the set of tentative covering constraints. Let $\mathcal{L}$ be a waiting list that stores triples $(\hat{\mathbf{w}}, z^R(\hat{\mathbf{x}}), \hat{\psi})$, where $\hat{\mathbf{w}}$ is a defense vector that must be revisited, $z^R(\hat{\mathbf{x}})$ is the real damage for an attack $\hat{\mathbf{x}} \in \mathcal{X}(\hat{\mathbf{w}})$ that we guess is critical, and $\hat{\psi}$ is the corresponding covering constraint. Algorithm 17 formally states the accelerated backward sampling algorithm. If $\mathcal{W}(\mathcal{C} \cup \mathcal{C}^\psi) \neq \emptyset$, then line 4 selects a defense $\hat{\mathbf{w}} \in \mathcal{W}(\mathcal{C} \cup \mathcal{C}^\psi)$ and lines 5–21 explore problem $\mathcal{Q}(\hat{\mathbf{w}})$ as in Algorithm 16. When $\hat{\mathbf{x}}$ has not been shown to be critical, line 22 computes the ratio $(\bar{z} - LB_i)/\bar{z}$, assuming that $\bar{z} > 0$, to measure the percent reduction to $\bar{z}$ that could be achieved by continuing to solve $\mathcal{Q}(\hat{\mathbf{w}})$. If this ratio is not greater than some parameter $\epsilon > 0$, then lines 23–24 store $(\hat{\mathbf{w}}, z^R(\hat{\mathbf{x}}), \mathbf{w}^\intercal \hat{\mathbf{x}} \geq 1)$ in $\mathcal{L}$, add the corresponding tentative covering constraint to $\mathcal{C}^\psi$, and stop the exploration of the current $\hat{\mathbf{w}}$. When $\mathcal{W}(\mathcal{C} \cup \mathcal{C}^\psi) = \emptyset$, if $\mathcal{C}^\psi \neq \emptyset$, then lines 29–38 reconsider the items stored in the waiting list. The first for-loop (in lines 29–33) iterates over $\mathcal{L}$ and moves from $\mathcal{C}^\psi$ to $\mathcal{C}$ all the covering constraints corresponding to attacks with $z^R(\hat{\mathbf{x}}^k) > \bar{z}$, discarding the associated $\mathbf{w}^k$ from further exploration. Note that if $z^R(\hat{\mathbf{x}}^k) = \bar{z}$, then we cannot yet discard $\mathbf{w}^k$: even if $\bar{z} = z^*$, the algorithm might not have updated the incumbent $(\bar{\mathbf{w}}, \bar{\mathbf{x}}, \bar{\mathbf{y}})$. The second for-loop (in lines 34–38) iterates over the remaining items in $\mathcal{L}$ and resumes exploration for any $\mathbf{w}^k$ that is still in $\mathcal{W}(\mathcal{C})$, but with $\epsilon = 0$. Finally, line 39 discards the remaining constraints in $\mathcal{C}^\psi$, empties the waiting list, and returns to the main while-loop.

Algorithm 17: Backward sampling framework with waiting list

Input : Problem $\mathcal{P}$ and a threshold parameter $\epsilon > 0$
Output: An optimal solution to $\mathcal{P}$

Initialize $\bar{z} = \infty$, covering constraints sets $\mathcal{C} = \emptyset$, $\mathcal{C}^\psi = \emptyset$, and waiting list $\mathcal{L} = \emptyset$;
Select $\hat{\mathcal{Y}}^1 \subseteq \mathcal{Y}$ as a sampling of the third-stage solution space and set counter $i = 0$;
while $\mathcal{W}(\mathcal{C} \cup \mathcal{C}^\psi) \neq \emptyset$ **do**
    Select $\hat{\mathbf{w}} \in \mathcal{W}(\mathcal{C} \cup \mathcal{C}^\psi)$;
    Initialize $UB_i = \infty$ and $LB_i = -\infty$;
    while $\mathrm{LB}_i < \bar{z}$ **do**
        Set $i = i + 1$;
        Solve $UB_i = z^I(\mathbf{w}, \hat{\mathcal{Y}}^i) = \max_{\mathbf{x} \in \mathcal{X}(\mathbf{w})} \min_{\mathbf{y} \in \hat{\mathcal{Y}}^i(\mathbf{x})} f(\mathbf{y})$ and obtain an optimal solution $(\hat{\mathbf{x}}, \hat{\mathbf{y}})$
        Solve $LB_i = z^R(\hat{\mathbf{x}}) = \min_{\mathbf{y} \in \mathcal{Y}(\hat{\mathbf{x}})} f(\mathbf{y})$ and obtain an optimal solution $\hat{\mathbf{y}}^*$;
        Set $\hat{\mathcal{Y}}^{i+1} = \hat{\mathcal{Y}}^i \cup \{\hat{\mathbf{y}}^*\}$;
        if $\mathrm{UB}_i < \bar{z}$ **then**
            Update global upper bound $\bar{z} \leftarrow UB_i$;
            Remove from $\hat{\mathcal{Y}}^{i+1}$ all solutions having objective value greater than UB_i;
            Add to $\hat{\mathcal{Y}}^{i+1}$ a subset of new solutions in $\mathcal{Y}_{UB_i}$;
        end
        if $\mathrm{LB}_i \geq \bar{z}$ **then**
            Add the covering constraint $\mathbf{w}^\intercal \hat{\mathbf{x}} \geq 1$ to $\mathcal{C}$;
        end
        if $\mathrm{LB}_i = \mathrm{UB}_i = \bar{z}$ **then**
            Update the incumbent solution $(\bar{\mathbf{w}}, \bar{\mathbf{x}}, \bar{\mathbf{y}}) \leftarrow (\hat{\mathbf{w}}, \hat{\mathbf{x}}, \hat{\mathbf{y}})$;
        end
        if $(\bar{z} - \mathrm{LB}_i)/\bar{z} \leq \epsilon$ *and* $\mathrm{LB}_i < \bar{z}$ **then**
            Add $(\hat{\mathbf{w}}, z^R(\hat{\mathbf{x}}), \mathbf{w}^\intercal \hat{\mathbf{x}} \geq 1)$ to the waiting list $\mathcal{L}$;
            Add the covering constraint $\mathbf{w}^\intercal \hat{\mathbf{x}} \geq 1$ to $\mathcal{C}^\psi$ and go to line 4;
        end
    end
end
if $\mathcal{C}^\psi \neq \emptyset$ **then**
    for $(\mathbf{w}^k, z^R(\hat{\mathbf{x}}^k), \psi^k) \in \mathcal{L}$ **do**
        if $z^R(\hat{\mathbf{x}}^k) > \bar{z}$ **then**
            A
        end
        dd ψ^k to $\mathcal{C}$, remove ψ^k from $\mathcal{C}^\psi$, and remove $(\mathbf{w}^k, z^R(\hat{\mathbf{x}}^k), \psi^k)$ from $\mathcal{L}$;
    end
    for $(\mathbf{w}^k, z^R(\hat{\mathbf{x}}^k), \psi^k) \in \mathcal{L}$ **do**
        if $\mathbf{w}^k \in \mathcal{W}(\mathcal{C})$ **then**
            R
        end
        esume solving $\mathcal{Q}(\mathbf{w}^k)$ with a threshold $\epsilon = 0$;
    end
    Reset $\mathcal{C}^\psi \leftarrow \emptyset$, $\mathcal{L} \leftarrow \emptyset$, and go to line 3;
end
Return $(\bar{\mathbf{w}}, \bar{\mathbf{x}}, \bar{\mathbf{y}})$;

Shortest Path Interdiction Problem with Fortification. The SPIF is formally defined on a directed graph $\mathcal{G} = (\mathcal{N}, \mathcal{A})$, where $\mathcal{N}$ is the set of nodes and $\mathcal{A} \subseteq \mathcal{N} \times \mathcal{N}$ is the set of arcs, s is the source node, and t is the destination node. For each arc $(i, j) \in \mathcal{A}$ there are two nonnegative attributes: the cost $c_{ij} \geq 0$ of traversing the arc, and the delay (or penalty) $d_{ij} \geq 0$ incurred when traversing an interdicted arc (so that crossing an interdicted arc costs $c_{ij} + d_{ij}$). Let $\mathbf{w}$ be the fortification decision variables defined over the arcs, where $\mathcal{W} \equiv \left\{\mathbf{w} : \mathbf{e}^\intercal \mathbf{w} \leq Q,\ \mathbf{w} \in \{0,1\}^{|\mathcal{A}|}\right\}$ enforces a cardinality constraint on the number of fortified arcs and ensures that the variables are binary. Similarly, let $\mathbf{x} \in \mathcal{X}(\mathbf{w})$ be the second-stage attack decision variables, where $\mathcal{X}(\mathbf{w}) \equiv \left\{\mathbf{x} : \mathbf{e}^\intercal \mathbf{x} \leq B,\ x_{ij} \leq 1 - w_{ij}\ \forall (i,j) \in \mathcal{A},\ \mathbf{x} \in \{0,1\}^{|\mathcal{A}|}\right\}$ ensures that a maximum of B unprotected arcs are attacked, and forces the $\mathbf{x}$-variables to be binary. Finally, let $\mathbf{y}$ be the third-stage arc-flow variables. The SPIF can be formally stated as:

$$\min_{\mathbf{w}\in\mathcal{W}} \max_{\mathbf{x}\in\mathcal{X}(\mathbf{w})} \min \sum_{(i,j)\in\mathcal{A}} (c_{ij} + d_{ij}x_{ij})y_{ij} \tag{82}$$

$$\text{s.t.} \sum_{\{j|(i,j)\in\mathcal{A}\}} y_{ij} - \sum_{\{j|(j,i)\in\mathcal{A}\}} y_{ji} = \begin{cases} 1, & \text{for } i = s \\ 0, & \text{for } i \in \mathcal{N}\backslash\{s,t\} \\ -1, & \text{for } i = t \end{cases} \tag{83}$$

$$y_{ij} \geq 0, \quad \forall (i,j) \in \mathcal{A}, \tag{84}$$

where in the objective function (82), the original cost of any arc is increased by d_{ij} when the arc is attacked (i.e., $x_{ij} = 1$). Constraints (83) define the shortest path flow conservation constraints, and (84) restrict the $\mathbf{y}$-variables to be nonnegative.

The implementation of the backward sampling framework for the SPIF requires a sampling scheme, an algorithm for solving two-level shortest path interdiction problems restricted over a sample of s-t paths, and a method to solve third-stage shortest path problems. The latter is simply accomplished via Dijkstra's algorithm. We discuss the first two components of our approach in the following subsections.

We adapt the *pulse algorithm* for the constrained shortest path problem to sample s-t paths from $\mathcal{G}$. The pulse algorithm conducts a recursive implicit enumeration of the solution space, supported by pruning strategies that efficiently discard a vast number of suboptimal solutions. The algorithm conducts a depth-first search beginning at s. When a partial path is pruned or the search reaches node t, the algorithm backtracks and continues the search through unexplored regions of the solution space.

We implemented two pruning strategies: *bound* and *arc-usage* pruning. The bound pruning strategy discards any path whose cost exceeds the current upper bound $\bar{z}$. To do so, we first obtain the minimum cost needed to reach node t from any node i, denoted by $\underline{c}_{it}$. Then, we prune any partial path from node s to node i with cost c_{si}, such that $c_{si} + \underline{c}_{it} > \bar{z}$.

In the arc-usage pruning strategy, we define an upper limit $\bar{u}$ on the number of paths in $\hat{\mathcal{Y}}$ that can use any arc (i, j). Let u_{ij} be the number of paths in $\hat{\mathcal{Y}}$ that use arc (i, j). When the search reaches node t, we add an s-t path to $\hat{\mathcal{Y}}$ and increase u_{ij} by one, for each arc (i, j) traversed in the path. Once $u_{ij} = \bar{u}$, we eliminate arc (i, j), forcing the pulse algorithm to explore paths that do not traverse arc (i, j). This strategy yields a diverse sample of s-t paths, which is desirable in our

backward sampling framework. Finally, we stop the sampling procedure once a maximum sample size limit is reached or once a time limit is exceeded.

We formulate the restricted problem $\mathcal{Q}(\hat{\mathbf{w}}, \hat{\mathcal{Y}})$ as a MIP. Let $\mathcal{P}^k$ be the set of arcs corresponding to the k^{th} path in sample $\hat{\mathcal{Y}}$, and let $c(\mathcal{P}^k)$ denote its cost. We formulate $\mathcal{Q}(\hat{\mathbf{w}}, \hat{\mathcal{Y}})$ as follows:

$$\max \quad z \tag{85}$$

$$\text{s.t.} \quad z \leq c(\mathcal{P}^k) + \sum_{(i,j)\in\mathcal{P}^k} d_{ij}x_{ij}, \quad \forall \mathcal{P}^k \in \hat{\mathcal{Y}}, \tag{86}$$

$$\mathbf{x} \in \mathcal{X}(\hat{\mathbf{w}}). \tag{87}$$

The objective function (85) maximizes z, which is constrained by (86) to be no more than the least cost path in $\hat{\mathcal{Y}}$, after considering delays caused by arc interdiction. Finally, constraints (87) ensure that we only consider feasible attacks in $\mathcal{X}(\hat{\mathbf{w}})$.

Observe that if our algorithm generates an attack $\hat{\mathbf{x}} \in \mathcal{X}(\hat{\mathbf{w}})$ having a perceived damage greater than $\bar{z}$, then $\bar{z}$ cannot be improved in the current iteration. In this case, our algorithm does not utilize the precise perceived damage value (beyond establishing that it exceeds $\bar{z}$). It is thus not necessary to optimize model (85)–(87) if we have proven that its objective exceeds $\bar{z}$, and so we add the objective target constraint $z \leq \bar{z} + \delta$, for a small constant $\delta > 0$, to model (85)–(87). This ensures that any attack $\hat{\mathbf{x}} \in \mathcal{X}(\hat{\mathbf{w}})$ with perceived damage strictly greater than $\bar{z}$ is sufficient to allow the overall algorithm to continue, even though $\hat{\mathbf{x}}$ may not optimize $\mathcal{Q}(\hat{\mathbf{w}}, \hat{\mathcal{Y}})$.

Furthermore, because the $\mathbf{x}$-variables are binary-valued and $d_{ij} \geq 0$, $\forall (i,j) \in \mathcal{A}$, the addition of the objective target constraint allows us to revise (86) as follows, where $(\bullet)^+ = \max\{0, \bullet\}$:

$$z \leq c(\mathcal{P}^k) + \sum_{(i,j)\in\mathcal{P}^k} \min\{d_{ij}, (\bar{z} + \delta - c(\mathcal{P}^k))^+\}x_{ij}, \quad \forall \mathcal{P}^k \in \hat{\mathcal{Y}}. \tag{88}$$

Constraints (88) are at least as tight as (86). (Note that (88) corresponding to some $\mathcal{P}^k$ may persist in our interdiction model for a few iterations after $\bar{z} + \delta \leq c(\mathcal{P}^k)$. We therefore require the coefficients of the $\mathbf{x}$-variables to be nonnegative in order to ensure the validity of (88).)

Capacitated Lot Sizing Interdiction Problem with Fortification. The capacitated lot sizing problem (CLSP) is a well-known NP-hard problem in which a facility manufactures a single product to satisfy a known demand over a finite planning horizon subject to production capacity constraints. In the CLSIF production capacity at any time period could be lost (e.g., due to machine failures). The system operator can ensure that capacity is protected against loss for some time periods (e.g., by performing preventive maintenance). In this context, an "attack" is not necessarily due to a malicious adversary, but represents some bounded worst-case scenario on capacity loss.

Formally, we define the CLSIF as the problem of finding a subset of time periods to fortify, in order to minimize the total cost resulting from a worst-case attack that disables production capacity on some of the unprotected time periods. Let $\mathcal{T}$ be the set of time periods in the planning horizon. For each time period $t \in \mathcal{T}$, let d_t be the demand, C_t be the production capacity, and let c_t, f_t, h_t, and

q_t be the production, setup, holding, and shortage cost, respectively. All parameters are assumed to be nonnegative.

Let $\mathbf{w} \in \mathcal{W}$ be the fortification decision variables and $\mathbf{x} \in \mathcal{X}(\mathbf{w})$ be the attack decision variables, where $\mathcal{W} \equiv \left\{\mathbf{w} : \mathbf{e}^\intercal \mathbf{w} \leq Q,\ \mathbf{w} \in \{0,1\}^{|\mathcal{T}|}\right\}$ establishes the defender's budget and ensures that the fortification variables are binary, and $\mathcal{X}(\mathbf{w}) \equiv \left\{\mathbf{x} : \mathbf{e}^\intercal \mathbf{x} \leq B,\ x_t \leq 1 - w_t\ \forall t \in \mathcal{T},\ \mathbf{x} \in \{0,1\}^{|\mathcal{T}|}\right\}$ ensures that a maximum of B unprotected time periods are attacked, and forces the attacker variables to be binary. Finally, let $\mathbf{y}$, $\mathbf{v}$, $\mathbf{I}$, and $\mathbf{s}$ be the third-stage decision variables modeling production, setup, inventory, and shortage, respectively. The CLSIF can be formally stated as:

$$\min_{\mathbf{w}\in\mathcal{W}} \max_{\mathbf{x}\in\mathcal{X}(\mathbf{w})} \min \quad \sum_{t\in\mathcal{T}} c_t y_t + f_t v_t + h_t I_t + q_t s_t \tag{89}$$

$$\text{s.t.} \quad I_0 = y_0 + s_0 - d_0, \tag{90}$$

$$I_t = I_{t-1} + y_t + s_t - d_t, \qquad \forall t \in \mathcal{T}\backslash\{0\}, \tag{91}$$

$$y_t \leq C_t v_t, \qquad \forall t \in \mathcal{T}, \tag{92}$$

$$v_t \leq 1 - x_t, \qquad \forall t \in \mathcal{T}, \tag{93}$$

$$y_t, I_t, s_t \geq 0, \qquad \forall t \in \mathcal{T}, \tag{94}$$

$$v_t \in \{0,1\}, \qquad \forall t \in \mathcal{T}. \tag{95}$$

The objective function (89) minimizes the total cost after interdiction. Constraints (90)–(91) are inventory constraints, constraints (92) enforce production capacity limits, and constraints (93) forbid production on interdicted time periods. Constraints (94) and (95) place bounds and logical restrictions on the decision variables.

In the following we discuss the three components required for solving the CLSIF: a sampling scheme, an approach for solving two-level CLSP interdiction problems restricted over a sample of third-stage solutions, and a method to solve third-stage CLSP problems.

Let $\mathcal{S}$ denote a production plan (third-stage recourse solution) that specifies values for $\mathbf{y}$, $\mathbf{v}$, $\mathbf{I}$, and $\mathbf{s}$. To obtain a sample of production plans, we propose a simple random search that iteratively generates a random attack plan $\mathbf{x}^r$, and solves a MIP to compute the optimal recourse response given $\mathbf{x}^r$. In particular, $\mathbf{x}^r$ interdicts K time periods randomly selected among $\{0,\ldots,|\mathcal{T}|\}$. We then solve the following MIP given $\mathbf{x}^r$:

$$\min_{(\mathbf{y},\mathbf{v},\mathbf{I},\mathbf{s})\in\mathcal{Y}(\mathbf{x}^r)} \quad \sum_{t\in\mathcal{T}} c_t y_t + f_t v_t + h_t I_t + q_t s_t, \tag{96}$$

where $\mathcal{Y}(\mathbf{x}^r)$ is the third-stage feasible region defined by inserting $\mathbf{x}^r$ in constraints (90)–(95).

Let production plan $\mathcal{S}^* = \{\mathbf{y}^*, \mathbf{v}^*, \mathbf{I}^*, \mathbf{s}^*\}$ be an optimal solution to the MIP given an attack plan $\mathbf{x}^r$, and let $c(\mathcal{S}^*)$ denote its cost. If $c(\mathcal{S}^*) \leq \bar{z}$, then we add $\mathcal{S}^*$ to the sample, and otherwise we discard $\mathcal{S}^*$. We repeat this procedure for a prescribed number of iterations (regardless of how many production plans are added to the sample). Note that the repeated solution of MIPs in the sampling phase of this algorithm may ultimately be too computationally intensive to justify its use. We will demonstrate that the solution of MIPs in this phase is justified. However, an alternative to this scheme would simply generate heuristic recourse solutions in response to randomly sampled

attacks. The tradeoff thus involves the quality of sampled solutions (where higher quality samples tend to speed overall convergence) versus the time required to generate them.

As in the SPIF, we formulate the restricted problem $\mathcal{Q}(\hat{\mathbf{w}}, \hat{\mathcal{Y}})$ as a MIP. Let $\mathcal{S}^k = \{\mathbf{y}^k, \mathbf{v}^k, \mathbf{I}^k, \mathbf{s}^k\}$ denote production plan k in $\hat{\mathcal{Y}}$ and $\mathcal{T}(\mathcal{S}^k) \equiv \{t \in \mathcal{T} \mid y_t^k > 0\}$ be the set of time periods in which plan $\mathcal{S}^k$ produces a positive amount of items. We formulate $\mathcal{Q}(\hat{\mathbf{w}}, \hat{\mathcal{Y}})$ analogously to (85)–(87):

$$\max \quad z \tag{97}$$

$$\text{s.t.} \quad z \leq c(\mathcal{S}^k) + \sum_{t \in \mathcal{T}(\mathcal{S}^k)} M_t^k x_t, \quad \forall \mathcal{S}^k \in \hat{\mathcal{Y}}, \tag{98}$$

$$\mathbf{x} \in \mathcal{X}(\hat{\mathbf{w}}). \tag{99}$$

We use a suitably large cost M_t^k to penalize attacked production plans. To determine this cost, we decompose $\mathbf{y}^k$ into values $a_{tt}^k, \ldots, a_{t|\mathcal{T}|}^k$, $\forall t \in \mathcal{T}$, where a_{tj}^k denotes the amount produced at period t that satisfies demand at period j, for $j \geq t$. One possible way of adjusting a solution if an attack occurs at period t is to simply retain the previous solution, but with $y_t^k = 0$. As a result, there will be a savings of $f_t + c_t y_t^k$ due to eliminated fixed and variable costs, plus any holding costs that were incurred due to production in period t. However, without adjusting production at any other period, we would incur additional shortage costs of $q_j a_{tj}^k$ for each $j \geq t$. Accordingly, we define the cost penalty for any production plan $\mathcal{S}^k$ at time period t as

$$M_t^k = \left(\sum_{j \in \mathcal{T}: j \geq t} q_j a_{tj}^k \right) - f_t - c_t y_t^k - \left(\sum_{j \in \mathcal{T}: j > t} \sum_{l=t}^{j-1} h_l a_{tj}^k \right). \tag{100}$$

Proposition 14 shows that (98) remains valid when M_t^k is defined as in (100).

Proposition 14 *Consider any $\mathbf{x} \in \mathcal{X}$ and let $\mathcal{S}^* \in \mathcal{Y}(\mathbf{x})$ be its corresponding optimal recourse response. For any production plan $\mathcal{S}^k$, we have that $c(\mathcal{S}^k) + \sum_{t \in \mathcal{T}(\mathcal{S}^k)} M_t^k x_t \geq c(\mathcal{S}^*)$, where the M-values are defined in (100).*

We use the objective target strategy introduced for the SPIF. Following the same logic, we add the constraint $z \leq \bar{z} + \delta$ to model (97)–(99), which allows us to tighten (98) as follows:

$$z \leq c(\mathcal{S}^k) + \sum_{t \in \mathcal{T}(\mathcal{S}^k)} \min\{M_t^k, (\bar{z} + \delta - c(\mathcal{S}^k))^+\} x_t, \quad \forall \mathcal{S}^k \in \hat{\mathcal{Y}}. \tag{101}$$

Calculating the real damage of an attack $\hat{\mathbf{x}}$ requires solving a CLSP in which the production capacity for time periods attacked by $\hat{\mathbf{x}}$ is set to zero. One simple approach solves the classical MIP model for the CLSP given attack plan $\hat{\mathbf{x}}$ stated in (96). Because the backward sampling framework does not require a specific solution method for the third-stage problem, we could employ any of the well-established methods for solving the CLSP, including the standard dynamic programming approach in which inventory at time t is used as state variable.

Computational Experiments. This section presents computational results on the SPIF (for the CLSIF, see Lozano and Smith (2015)). We assess the performance of our algorithm on the SPIF using randomly generated grid networks and real road networks. We coded our algorithm in Java, using Eclipse SDK version 4.4.1, and executed the experiments on a machine having an Intel Core i7–3537U CPU (two cores) running at 2.00 GHz with 2 GB of RAM allocated to the Java Virtual Machine memory heap on Windows 8. We impose a time limit of four hours (14,400s) and solve all mathematical optimization problems using Gurobi 5.6. All instances and source code used in this section are available from the authors.

We generate directed grid networks having a source node s, a sink node t, and $m \times n$ nodes arranged in a grid of m rows and n columns. There exists an arc from s to every node in the first column and an arc from every node in the last column to t. Also, arcs exist from each node in grid row r and column c to (existing) nodes in positions $(r+1, c)$, $(r-1, c)$, $(r, c+1)$, $(r+1, c+1)$, and $(r-1, c+1)$ provided that these are not vertical arcs in the first or last columns.

We build networks with sizes ranging from 10×10 to 60×60. For each network size we explore different (cost, delay) configurations in which arc costs (delays) are random integers uniformly distributed between $[1, c]$ ($[1, d]$), where c (d) denotes the maximum cost (delay). We explore the following (c, d) configurations: $(10, 5)$, $(10, 10)$, $(10, 20)$, $(100, 50)$, $(100, 100)$, and $(100, 200)$. For a fixed network size and (c, d) configuration, we generate ten instances with different random arc attributes for a total of $360 = 6 \times 6 \times 10$ different instances. For each instance we solve six problems with different Q values in $\{3, 4, 5, 7\}$ and B values in $\{3, 4, 5\}$, for a total of $2160 = 360 \times 6$ experiments. After tuning the algorithm parameters, we set the maximum sample size to 100, the sampling time limit to 1 second, the arc-usage upper limit to 20, threshold ϵ to 0.1, and δ to 1.

Tables 1 and 2 show the computational results for medium- and large-sized grid networks, respectively. The first five columns show grid size, number of nodes and arcs, and the defender's and attacker's budget (Q and B), respectively. For each of the six (c, d) configurations, the tables depict the average CPU time obtained over ten runs (Avg) and the largest CPU time obtained over those runs (Max).

Table 1 shows that on average, our algorithm finds optimal solutions for the 10×10 and 20×20 networks in just a few seconds, and requires less than one minute to solve the 30×30 networks, which have more than 4000 arcs. The maximum execution times are close to the average times; even in the worst case (30×30 grids with $Q = 7$, $B = 5$, and $(c, d) = (10, 20)$), the algorithm terminates in just over two minutes.

Table 1: Computational time in CPU seconds for solving the SPIF over medium-size grid networks

Instance	Nodes	Arcs	Q	B	Cost–delay configuration (c, d)											
					$(10, 5)$		$(10, 10)$		$(10, 20)$		$(100, 50)$		$(100, 100)$		$(100, 200)$	
					Avg	Max	Avg	Max	Avg	Max	Avg	Max	Avg	Max	Avg	Max
10×10	102	416	3	3	0.1	0.3	0.1	0.2	0.1	0.2	0.1	0.2	0.1	0.3	0.1	0.2
			4	3	0.1	0.1	0.1	0.2	0.1	0.2	0.1	0.2	0.1	0.2	0.2	0.2
			3	4	0.1	0.2	0.2	0.5	0.2	0.5	0.3	0.5	0.3	0.7	0.3	0.7
			5	4	0.2	0.3	0.3	0.5	0.3	0.6	0.3	0.4	0.4	0.7	0.4	0.8
			4	5	0.3	0.4	0.5	1.1	0.8	2.2	0.6	1.5	1.0	2.9	1.1	2.8
			7	5	0.7	1.0	0.9	1.6	1.0	2.3	0.8	1.2	1.3	2.9	1.3	1.9
20×20	402	1826	3	3	0.3	0.9	0.5	1.9	0.6	3.3	0.7	2.0	0.7	1.3	0.8	2.0
			4	3	0.4	1.2	0.6	2.2	0.8	4.3	0.9	2.3	0.8	1.6	0.9	2.0
			3	4	0.6	1.5	1.1	3.2	1.1	5.1	2.2	5.1	2.4	5.9	2.4	8.0
			5	4	1.0	2.7	2.2	10.8	2.2	12.8	2.1	4.4	2.9	6.1	2.9	9.1
			4	5	1.8	5.6	4.2	18.2	4.1	16.8	6.5	16.1	8.3	22.4	9.7	33.4
			7	5	3.6	10.8	5.7	13.1	6.7	28.3	7.3	13.3	9.8	21.3	12.5	36.8
30×30	902	4236	3	3	0.8	1.1	1.2	3.6	1.9	7.3	1.8	7.5	2.0	3.9	2.3	6.0
			4	3	0.9	1.2	1.4	3.7	2.0	6.9	2.3	11.2	2.6	6.1	2.4	5.7
			3	4	1.6	2.6	3.8	14.9	6.0	25.6	4.4	15.0	5.5	13.3	6.3	15.5
			5	4	2.7	4.1	6.1	27.4	8.1	36.4	6.2	21.5	10.0	31.0	10.2	36.9
			4	5	5.2	11.8	15.9	77.2	23.2	94.8	15.9	60.2	24.2	61.1	27.1	73.2
			7	5	11.7	22.7	26.7	108.2	30.1	131.7	21.9	62.7	35.8	101.1	36.7	91.2

Table 2: Computational time in CPU seconds for solving the SPIF over large-size grid networks

Instance	Nodes	Arcs	Q	B	Cost–delay configuration (c, d)											
					$(10, 5)$		$(10, 10)$		$(10, 20)$		$(100, 50)$		$(100, 100)$		$(100, 200)$	
					Avg	Max	Avg	Max	Avg	Max	Avg	Max	Avg	Max	Avg	Max
40×40	1602	7646	3	3	3.0	7.6	6.3	26.5	8.8	48.3	4.3	6.7	7.8	26.6	6.8	21.2
			4	3	3.5	8.1	7.5	35.2	9.6	51.4	5.2	8.2	9.3	37.6	7.9	26.7
			3	4	3.9	8.2	24.5	145.5	45.7	321.7	12.7	24.6	18.6	58.2	16.3	39.1
			5	4	6.8	12.9	34.2	196.6	54.0	376.6	16.0	31.4	28.3	127.2	24.4	91.8
			4	5	9.8	23.0	212.6	1786.3	194.0	1103.3	39.0	94.8	75.6	217.7	64.0	185.8
			7	5	23.4	61.7	211.4	1400.6	330.2	2153.0	56.9	101.2	181.4	1155.0	111.0	497.8
50×50	2502	12,056	3	3	6.7	16.7	10.2	18.5	15.6	43.0	19.3	45.6	38.3	141.7	47.6	130.1
			4	3	7.6	17.2	11.9	25.7	16.2	43.4	20.1	45.9	38.1	132.6	47.6	134.9
			3	4	10.5	50.1	30.8	126.3	35.7	190.2	28.0	61.8	114.6	717.4	80.7	166.0
			5	4	13.2	59.8	33.4	122.8	43.3	231.1	44.5	116.4	244.7	1979.0	117.2	444.0
			4	5	24.2	147.3	121.6	588.7	112.5	787.0	161.3	973.0	197.1	715.6	317.7	1049.9
			7	5	44.0	208.1	210.8	1355.7	142.1	847.1	344.8	1702.7	527.7	3614.9	375.1	1385.0
60×60	3602	17,466	3	3	13.3	23.2	24.3	53.9	38.8	64.7	29.5	46.6	60.6	230.6	82.3	236.8
			4	3	14.2	22.1	32.8	122.8	50.5	146.8	32.4	47.4	64.8	259.4	86.7	273.8
			3	4	17.0	32.2	64.5	272.4	79.8	291.9	34.6	52.5	107.2	339.8	149.7	512.6
			5	4	24.0	65.2	119.8	723.0	125.5	663.6	55.0	122.0	142.0	460.0	202.8	910.8
			4	5	35.6	67.5	223.5	1263.4	236.3	1070.4	155.9	578.8	1479.0	8103.9	1587.3	11,052.4
			7	5	81.2	273.9	346.0	1477.3	491.8	3382.5	317.1	1056.6	1575.9	8079.2	1733.4	11,177.9

Table 2 shows that, on average, the algorithm terminates in less than six minutes for the 40×40 networks, in less than nine minutes for the 50×50 networks, and in less than 29 minutes for the 60×60 networks, for any combination of c, d, Q, and B examined. The maximum execution times are larger relative to the average CPU times on these instances, and some of them require roughly four hours of computational time over the 60×60 networks. This behavior is expected, considering that these networks have more than 3000 nodes and 17,000 arcs. Table 2 suggests that the instances become more difficult as d grows larger relative to c and when the cost (delay) values increase (implying that $(c, d) = (100, 200)$ are typically the most challenging instances). Finally, an increase in the attacker's budget tends to have a greater impact on the computation time than an increase in the defender's budget.

We compare our approach (Sampling) to the current state-of-the-art algorithm by Cappanera and Scaparra (2011) over medium-sized instances, who graciously provided their code for the purposes of this comparison. Table 3 shows the results for this comparison. Here, the "Avg" column depicts the average CPU time in seconds, computed only among the instances solved within the time limit. As before, "Max" refers to maximum CPU seconds out of the 60 instances solved for the row, and "# solved" gives the number of instances solved within the four-hour time limit.

Table 3: Comparing the backward sampling algorithm to the state-of-the-art algorithm for SPIF

Instance	Nodes	Arcs	Q	B	Sampling			SPI		
					Avg	Max	# solved	Avg	Max	# solved
			3	3	0.1	0.3	60	1.9	4.6	60
10×10	102	416	5	4	0.3	0.8	60	30.9	162.8	60
			4	5	0.7	2.9	60	67.5	284.2	60
			3	3	0.6	3.3	60	26.7	305.1	60
20×20	402	1,826	5	4	2.2	12.8	60	1128.4	7723.4	60
			4	5	5.8	33.4	60	2495.2	>14,400	56
			3	3	1.7	7.5	60	766.9	12,728.8	60
30×30	902	4,236	5	4	7.2	36.9	60	4857.3	>14,400	45
			4	5	18.6	94.8	60	4256.8	>14,400	26

Table 3 shows that our algorithm compares favorably to SPI, consistently reducing computational time by more than two orders of magnitude both in terms of average and maximum execution times, for any combination of (Q, B) examined. Moreover, our sampling algorithm solves all instances within the time limit while SPI solves 56 instances when $(Q, B) = (4, 5)$ on the 20×20 networks, 45 instances when $(Q, B) = (5, 4)$ on the 30×30 networks, and 26 instances when $(Q, B) = (4, 5)$.

We now use the road networks from Washington (DC), Rhode Island (RI), and New Jersey (NJ) presented by Raith (2009). These networks range from 9559 nodes and 39,377 arcs to 330,386 nodes and 1,202,458 arcs. For each road network, there are nine randomly selected s-t pairs. We define c_{ij} as the arc distance and set $d_{ij} = 10{,}000,\ \forall (i, j) \in \mathcal{A}$. For each network and s-t pair we explore six budget configurations for a total of $162 = 3 \times 9 \times 6$ experiments. Table 4 shows the results for these experiments.

Table 4 shows that the algorithm solves all DC instances to optimality within the time limit. The average CPU time for these instances is less than nine minutes and the worst execution time is well under one hour. On the RI network, the algorithm solves all instances with $B \leq 4$ and solves

Table 4: Computational time in CPU seconds for solving the SPIF over road networks

Instance	Nodes	Arcs	Q	B	Avg	Max	# solved
DC	9559	39,377	3	3	45.8	124.9	9
			4	3	50.6	127.1	9
			3	4	92.2	402.9	9
			5	4	103.0	374.8	9
			4	5	492.8	2829.5	9
			7	5	450.1	1906.4	9
RI	53,658	192,084	3	3	284.5	756.0	9
			4	3	295.6	817.3	9
			3	4	800.7	4925.1	9
			5	4	946.2	5974.9	9
			4	5	560.1	>14,400	8
			7	5	754.0	>14,400	8
NJ	330,386	1,202,458	3	3	6743.8	10,551.9	9
			4	3	6345.8	>14,400	8
			3	4	6526.8	>14,400	8
			5	4	6964.3	>14,400	8
			4	5	7354.6	>14,400	8
			7	5	8452.6	>14,400	8

all but one instance each when $(Q, B) = (4, 5)$ and $(7, 5)$. Average CPU times are less than 15 minutes among the instances solved to optimality within the time limit, for any choice of (Q, B). On the NJ network, the algorithm solves all instances with $Q = B = 3$ and solves all but one instance in each set corresponding to the other (Q, B) combinations. Average times are roughly two hours among the instances solved to optimality.

9 Defender-Attacker Influence Spread Games

In the study of Hemmati, Smith, and Thai (2014), we consider a network whose nodes can be influenced somehow by competing agents. These networks may model social or geographical interactions among entities, and influence can represent the spread of rumors, infections, or other types of transmissions over the nodes. A key question related to competition over such networks seeks to identify vital nodes in the network with the aim of protecting them from being the source of influence propagation from competing agents.

We consider a scenario in which two players, a defender and an attacker, compete on a directed network $G(V, A)$, where V is the set of nodes and A is the set of arcs. Initially, the defender owns every node in the network, and can protect a subset of nodes against an impending action by the attacker. The attacker then acts, with full knowledge of the defender's action, to capture a set of unprotected nodes. For consistency with prior related studies, we say that captured nodes have been *influenced* by the attacker. This initial action takes place at time 0, and the game continues for T (discrete) time periods according to the following rules.

1. An influenced node remains influenced for the remainder of the time horizon.

2. A node that was protected by the defender cannot be influenced at any time.

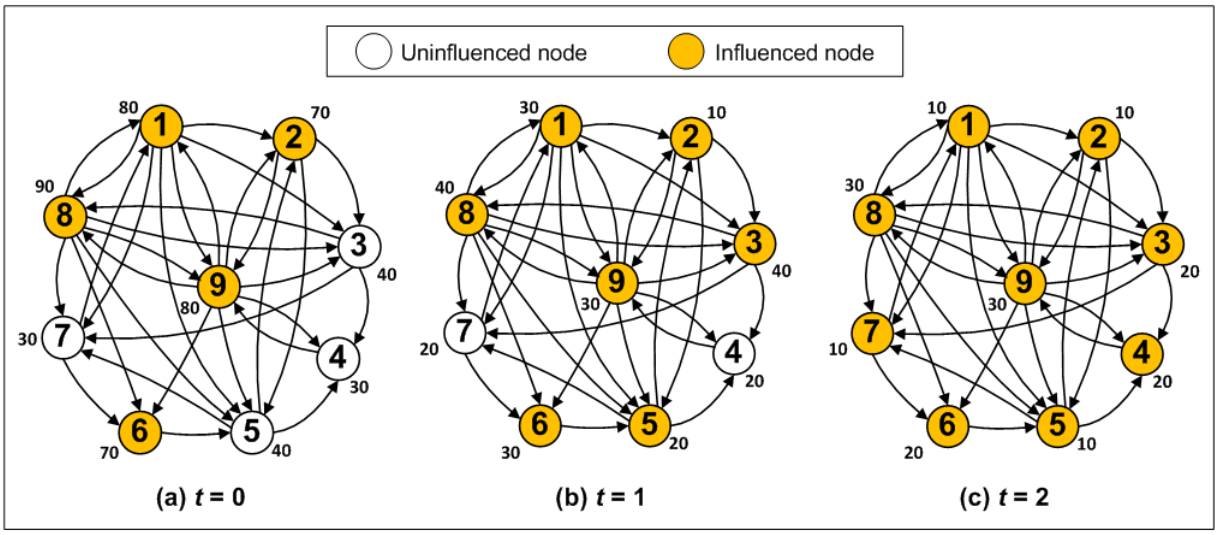


Figure 16: An instance with $Q = 3$ and $T = 2$, in the absence of protected nodes.

3. Consider an unprotected node $j \in V$ that is not influenced at time $t \in \{0, \ldots, T-1\}$. Then node $j \in V$ becomes influenced at time $t+1$ if and only if there are some Q nodes $i \in V$ such that i is influenced at time t, and $(i, j) \in A$.

4. The attacker earns a reward of r_i^t if node i is influenced at time t (but not at time $t-1$, if $t \geq 1$).

The goal of the defender is to minimize the maximum sum of rewards that the attacker can earn (e.g., minimizing the maximum amount of damage that the attacker could possibly inflict on the defender's network).

Figures 16 and 17 illustrate a problem instance in which $Q = 3$ and $T = 2$. The r-values are stated for each time period next to each node. Consider the case in which no nodes are initially protected, and the attacker influences nodes 1, 2, 6, 8, and 9 at $t = 0$ (Figure 16a). As the result of this action, nodes 3 and 5 become influenced at $t = 1$, because nodes 1, 2, and 8 are influenced at $t = 0$ (Figure 16b). Nodes 4 and 7 become influenced at $t = 2$ (Figure 16c). Hence, the attacker earns a reward of 480. Next, suppose that the defender protects nodes 6 and 9. Then an optimal response from the attacker is to influence nodes 1, 2, and 8 (Figure 17a). Although nodes 3 and 5 become influenced at $t = 1$ (Figure 17b), only node 7 will be influenced at $t = 2$ (Figure 17c). In this case, the attacker's reward reduces to 310.

For each node $i \in V$, define the set of incoming neighbors of i as $V^-(i) = \{j \in V : (j, i) \in A\}$, and the set of outgoing neighbors of i as $V^+(i) = \{j \in V : (i, j) \in A\}$. Let $\mathcal{T} = \{1, \ldots, T\}$ be the set of time periods. Recall that an unprotected node $i \in V$ that is not influenced at time $t-1$ becomes influenced at time $t \in \mathcal{T}$ if at least Q nodes in $V^-(i)$ are influenced at time $t-1$, where we refer to Q as the threshold influence parameter. Also, recall that the attacker earns a reward of r_i^t if node $i \in V$ is influenced at time $t \in \mathcal{T} \cup \{0\}$ for the first time, where $r_i^0 \geq \cdots \geq r_i^T$. There exists a cost of $c_i,\ i \in V$, for the attacker to influence node i at time zero. Similarly, the defender

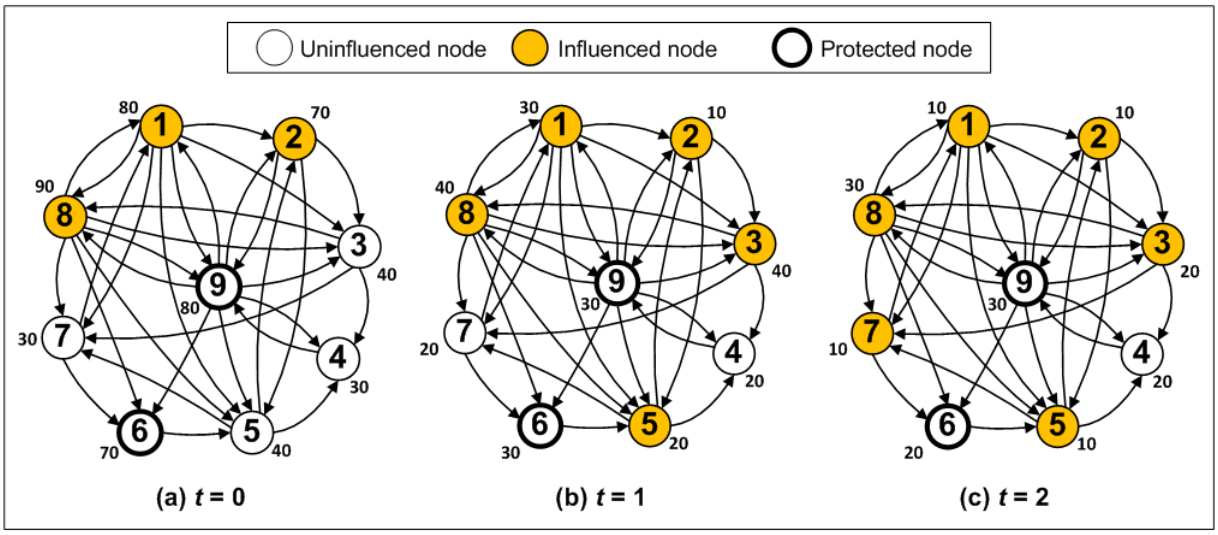


Figure 17: An instance with $Q = 3$ and $T = 2$, with nodes 6 and 9 protected by the defender.

incurs a cost of $b_i,\ i \in V$, to protect node i. In our model, the defender (attacker) has a budget of B (D) to protect (initially influence) nodes.

In order to formulate this problem, we first define two sets of binary decision variables $x_i,\ i \in V$, and y_i^0. In our model, $x_i = 1$ if the defender protects node $i \in V$, and $x_i = 0$ otherwise. Also, $y_i^0 = 1$ if node $i \in V$ is influenced by the attacker at time zero, and $y_i^0 = 0$ otherwise. Additionally, we introduce binary decision variables $y_i^t = 1,\ t \in \mathcal{T}$, if node $i \in V$ is influenced at time t, and $y_i^t = 0$ otherwise. Note that we have separated y^0-variables from y^t-variables to emphasize the difference between influence at $t = 0$ and $t > 0$, because the latter results from the spread of influence. The defender's problem can be formulated as follows.

$$\begin{aligned} \min \quad & z(x) && && (102a)\\ \text{s.t.} \quad & b^T x \le B && && (102b)\\ & x_i \in \{0,1\} && \forall i \in V, && (102c) \end{aligned}$$

where $z(x)$ is the optimal objective value of the attacker's problem, which can be computed by solving the following integer program given some fixed value of $x = \bar{x}$:

$$\begin{aligned} \textbf{ATT1}(\bar{x}) : z(\bar{x}) = \max \quad & \sum_{i \in V} \left(r_i^0 y_i^0 + \sum_{t=1}^{T} r_i^t (y_i^t - y_i^{t-1}) \right) && && (103a)\\ \text{s.t.} \quad & y_i^t \le 1 - \bar{x}_i && \forall i \in V,\ t \in \mathcal{T} \cup \{0\} && (103b)\\ & Q y_i^t \le Q y_i^0 + \sum_{j \in V^-(i)} y_j^{t-1} && \forall i \in V,\ t \in \mathcal{T} && (103c)\\ & c^T y^0 \le D && && (103d)\\ & y_i^0 \in \{0,1\} && \forall i \in V && (103e)\\ & y_i^t \in \{0,1\} && \forall i \in V,\ t \in \mathcal{T}. && (103f) \end{aligned}$$

The objective function (102a) reflects the defender's goal of minimizing the maximum reward earned by the attacker (computed by solving (103)). The defender's budget limit and the binariness of the x-variables are enforced by constraints (102b) and (102c), respectively. The objective function (103a) represents the attacker's reward, noting that $y_i^0 = 1$ if node $i \in V$ is initially influenced, and $y_i^t - y_i^{t-1} = 1$ if node $i \in V$ is influenced for the first time at time $t \in \mathcal{T}$. Constraint (103b) implies that a protected node can never be influenced by the attacker. Constraints (103c) governs the spread of influence: If node $i \in V$ is initially influenced, then the right-hand-side (RHS) of constraints (103c) will be at least Q for all $t \in \mathcal{T}$, implying that node i will remain influenced at all time periods. Now, suppose that node $i \in V$ is not initially influenced, and consider constraint (103c) for node i and time $t \in \mathcal{T}$. The constraint implies that node i can be influenced at time t if and only if $\sum_{j \in V^-(i)} y_j^{t-1} \geq Q$, i.e., if and only if at least Q nodes adjacent to node i are influenced at time $t-1$. Note that for any node-time pair $i \in V$ and $t \in \mathcal{T}$, y_i^t is present with a nonnegative coefficient in the objective function (due to nonincreasing values for node i's rewards over time). Therefore, there exists an optimal solution in which $y_i^t = 1$ if and only if either node i is initially influenced, or at least Q nodes adjacent to node i are influenced at time $t-1$. Finally, constraint (103d) enforces the attacker's budget limit, and constraints (103e) and (103f) restrict the y-variables to be binary-valued.

In the rest of this discussion, we define $X = \{x \in \{0,1\}^{|V|} : b^T x \leq B\}$ as the set of possible actions for the defender. Given $\bar{x} \in X$, we also define $Y(\bar{x}) = \{y^0 \in \{0,1\}^{|V|} : c^T y^0 \leq D,\ y_i^0 \leq 1 - \bar{x}_i,\ \forall i \in V\}$ as the set of available actions for the attacker at time zero when the defender chooses $\bar{x}$.

We may also wish to consider the case in which the influence threshold value depends on the node being influenced, and so node i becomes influenced at time t if some Q_i nodes in $V^-(i)$ are influenced at time $t-1$. This case can be transformed to the case in which all nodes have a common threshold value, Q. To see this, let $Q = \max_{i \in V}\{Q_i\}$. Create a set of Q dummy nodes that are impossible to protect and free for the attacker to initially influence, and let the reward for influencing these nodes equal 0 at all time periods. For each $i \in V$, create an arc from $Q - Q_i$ of the dummy nodes to node i. Because an optimal solution exists in which all of these dummy nodes would be initially influenced, only Q_i more nodes in $V^-(i)$ from the original graph must be influenced in order to influence node i, as desired. Hence, for simplicity, we use the common threshold value of Q in this paper. ■

We finish this section by observing that the attacker's problem is strongly NP-hard. To see this, we sketch a (polynomial) reduction from the *dominating set* problem to a variant of the attacker's problem. In the dominating set problem, we seek a subset D of nodes in an undirected graph $\bar{G}(\bar{V}, \bar{E})$ such that each node in $\bar{V} \backslash D$ is adjacent to at least one node in D, and such that $|D| \leq \delta$ for some given positive integer δ. Now, consider the attacker's problem with $Q = 1$ and $T = 1$. Let the attacker's problem network $G(\bar{V}, A)$ consist of the same node set as in the dominating set instance, and let A contain two directed arcs, (i,j) and (j,i), for each $(i,j) \in \bar{E}$. Also, define $B = \delta$ and $b_i = r_i^0 = r_i^1 = 1,\ \forall i \in \bar{V}$. There exists a dominating set having δ nodes if and only if there exists a solution to the attacker's problem with reward $|\bar{V}|$. Hence, the attacker's problem is strongly NP-hard. Moreover, the defender's problem is also NP-hard, because evaluating $z(x)$ cannot be done in polynomial time unless P = NP.

Exact Solution Method. We now provide a cutting-plane scheme to solve the problem considered here. The inherent difficulty in solving model (102) is due to the nonconvexity of the attacker's problem, which prohibits us from readily obtaining a strong (minimization) dual to model (103) and employing standard interdiction models. In order to devise a cutting-plane algorithm, we start by proposing a reformulation of the foregoing problem.

We reformulate (102) by introducing a variable z, and minimizing z subject to the restriction that $x \in X$ and $z \geq z(x)$. We refer to a pair $(\bar{x}, \bar{z})$ as a *two-stage feasible solution* if $\bar{x} \in X$ and $\bar{z} \geq z(\bar{x})$. Defining Ω as the set of all two-stage feasible solutions, we obtain the following reformulation for the defender's problem:

$$\textbf{DEF}: \quad \min \ z \tag{104a}$$
$$\text{s.t.} \ x \in X \tag{104b}$$
$$(x, z) \in \Omega. \tag{104c}$$

Note that an optimal solution $(x^\star, z^\star)$ to (104) satisfies $z^\star = z(x^\star)$, because (104) is a minimization program.

Let $\bar{\Omega} \supseteq \Omega$ be a feasible region induced by a set of affine inequalities, and define DEF-R as the relaxation of (104) obtained by replacing Ω with $\bar{\Omega}$. The motivation for introducing DEF-R stems from the fact that exponentially many inequalities may be required for the explicit definition of Ω. Hence, our approach starts with an initial $\bar{\Omega}$ defined by a small (polynomial-size) set of inequalities. If an optimal solution $(\bar{x}, \bar{z})$ to DEF-R is two-stage feasible, then it must be also optimal to (102), because DEF-R is a relaxation of (102). Otherwise, we can augment DEF-R with a cutting plane (as discussed below) and re-solve DEF-R in an iterative fashion until a two-stage feasible solution is found. We start by computing lower and upper bounds for the optimal objective value of the attacker's problem.

Lemma 12 *Let $i_{(j)}$, $1 \leq j \leq |V|$, be the node having the j^{th} largest reward at $t = 0$. Denote by q_1 the largest integer such that $\sum_{i \in V'} c_i \leq D$, $\forall V' \subseteq V : |V'| = q_1$. Also, denote by p_2 the largest integer such that $\sum_{i \in V'} b_i \leq B$ for some $V' \subseteq V : |V'| = p_2$. Define:*

$$z_{\min} = \sum_{j=p_2+1}^{\min\{|V|,\, p_2+q_1\}} r^0_{i_{(j)}}.$$

We have:

$$z(x) \geq z_{\min} \qquad \forall x \in X. \tag{105}$$

The proof of this and other lemmas are contained in Hemmati et al. (2014).

Lemma 13 *Let q_2 be the largest integer such that $\sum_{i \in V'} c_i \leq D$ for some $V' \subseteq V : |V'| = q_2$. Given a defender's decision vector $\bar{x}$, an upper bound on the attacker's optimal objective value is obtained by solving the following problem.*

$$z_{\max}(\bar{x}) = \max \ \sum_{i \in V} (1 - \bar{x}_i) \left(r^0_i y^0_i + r^1_i (1 - y^0_i) \right) \tag{106a}$$

$$\text{s.t.} \quad \sum_{i \in V} y_i^0 \leq q_2 \tag{106b}$$

$$0 \leq y_i^0 \leq 1 \qquad \forall i \in V \tag{106c}$$

Note that problem (106) can be optimized in $\mathcal{O}(|V| \log(|V|))$ steps by sorting the $(1-\bar{x}_i)(r_i^0 - r_i^1)$-values in nonincreasing order, and setting $y_i^0 = 1$ for each node $i \in V$ corresponding to the q_2-largest such coefficients.

The bound $z(\bar{x}) \leq z_{\max}(\bar{x})$ is valid for any $\bar{x} \in X$, and so one strategy may enumerate several candidate solutions $\bar{x} \in X$, compute $z_{\max}(\bar{x})$ for each vector, and obtain the minimum such value as a valid upper bound. Additionally, we can solve the following optimization problem.

$$z_{\max} = \min \;\; z_{\max}(x) \tag{107a}$$

$$\text{s.t.} \;\; x \in X. \tag{107b}$$

Observe that (107) is a two-stage program in which the feasible region of the inner problem is independent of x-variables. This allows us to state (107) as a linear mixed-integer program by dualizing the inner problem. Let α and β_i, $i \in V$, be the dual variables corresponding to constraints (106b) and (106c), respectively. We obtain the following reformulation of (107).

$$z_{\max} = \sum_{i \in V} r_i^1 + \min \;\; q_2 \alpha + \sum_{i \in V} \left(\beta_i - r_i^1 x_i\right) \tag{108a}$$

$$\text{s.t.} \;\; \alpha + \beta_i + (r_i^0 - r_i^1) x_i \geq (r_i^0 - r_i^1) \qquad \forall i \in V \tag{108b}$$

$$\alpha \geq 0 \tag{108c}$$

$$\beta_i \geq 0 \qquad \forall i \in V \tag{108d}$$

$$x \in X. \tag{108e}$$

Note that problem (108) has to be solved only once in order to obtain an upper bound for the problem, and hence will not likely represent a substantial portion of the time required to solve the overall model we investigate here.

An alternative strategy is to heuristically select some $\bar{x} \in X$, e.g., by using the following greedy algorithm. Initialize $\bar{x}_i = 0$, $\forall i \in V$, and set a "remaining budget" value $\bar{B} = B$. Find an index i such that r_i^0 is maximized over all $i \in V$ such that $\bar{x}_i = 0$ and b_i is not more than $\bar{B}$. If no such index i exists, then set the upper bound to $z_{\max}(\bar{x})$. Else, set $\bar{x}_i = 1$, reduce $\bar{B}$ by b_i, and reiterate. In our initial computational experiments, we compare the effectiveness of the exact formulation (108) versus the use of this greedy algorithm, and employ the most effective one in our computational study. ■

Cutting-plane Algorithm. We now provide valid inequalities for DEF-R, and we propose a cutting-plane scheme for identifying an optimal solution to (102).

Consider $\bar{x} \in X$ and suppose that $\bar{y} = (\bar{y}^0, \ldots, \bar{y}^T)$ is optimal to ATT1($\bar{x}$). We define $\bar{\tau}_i$, $i \in V$, as the earliest time that node i is influenced in the solution $\bar{y}$. We use the convention $\bar{\tau}_i = T + 1$ if node $i \in V$ is never influenced by the attacker, and we let $r_i^{T+1} = 0$. Consider the vector

$\bar{\tau} = (\bar{\tau}_1, \ldots, \bar{\tau}_{|V|})$, and for all $i \in V$, define $R_i^{\bar{\tau}}$ as the set of all unprotected nodes $j \in V$ such that there exists a directed path from node i to node j using $\bar{\tau}_j$ or fewer arcs in A (and hence, $i \in R_i^{\bar{\tau}}$).

Lemma 14 *Consider a solution $\bar{x} \in X$ in which $\bar{x}_i = 0$ for some node $i \in V$. Let $\bar{y} = (\bar{y}^0, \ldots, \bar{y}^T)$ be an optimal solution to ATT1($\bar{x}$), with corresponding vector $\bar{\tau}$. Suppose that the solution $\hat{x}$, which is identical to $\bar{x}$ with the exception of setting $\hat{x}_i = 1$, is feasible to* (102)*. Then we have:*

$$z(\hat{x}) \geq z(\bar{x}) - \sum_{j \in R_i^{\bar{\tau}}} r_j^{\bar{\tau}_j}. \tag{109}$$

For any given $\bar{x} \in X$, define $P_{\bar{x}}$ as the set of protected nodes in $\bar{x}$. Furthermore, let $\bar{y} = (\bar{y}^0, \ldots, \bar{y}^T)$ be an optimal solution to ATT1($\bar{x}$), and define $V_{\bar{x}}$ as the set of all influenced nodes in solution $\bar{y}$. We introduce our first valid inequality for (104) in the next theorem.

Theorem 16 *Let $(\bar{x}, \tilde{z})$ be an optimal solution to DEF-R, and suppose that $\tilde{z} < z(\bar{x})$. Let $\bar{y}$ be an optimal solution to ATT1($\bar{x}$), with corresponding vector $\bar{\tau}$. The following inequality:*

$$z \geq z(\bar{x}) - \sum_{i \in V_{\bar{x}}} \left(\min \left\{ z(\bar{x}) - z_{\min}, \sum_{j \in R_i^{\bar{\tau}}} r_j^{\bar{\tau}_j} \right\} \right) x_i, \tag{110}$$

is valid to (104) *and cuts off $(\bar{x}, \tilde{z})$.*

The following cutting-plane algorithm, which we call CPA, is then given as follows.

Step 0 (Initialization). Set the upper bound $UB = z_{\max}$ and the lower bound $LB = z_{\min}$. Let $\bar{\Omega} = \{(x, z) : x \in X,\ z \geq z_{\min}\}$.

Step 1 (Lower Bound). Identify an optimal solution $(\bar{x}, \tilde{z})$ to DEF-R. Set $LB = \tilde{z}$, and proceed to Step 2.

Step 2 (Upper Bound). Solve ATT1($\bar{x}$). If $z(\bar{x}) < UB$, then set $UB = z(\bar{x})$, and let $\bar{x}$ be the incumbent solution to (102). In either case, proceed to Step 3.

Step 3 (Termination/Cut Routine). If $LB = UB$, then terminate with the incumbent solution being optimal. Otherwise, add (110) to $\bar{\Omega}$, and return to Step 1.

At each iteration, CPA either terminates with an optimal solution in Step 3, or the identified solution $(\bar{x}, \tilde{z})$ in Step 1 will be cut off by the inequality added in Step 3. Note that X is a finite set, and that if some solution x' is encountered as the optimal solution in Step 1 in two different iterations k_1 and k_2 of CPA, then $LB = UB$ after iteration k_2. This behavior is due to the fact that inequality (110) was added for $x = x'$ after iteration k_1; hence, at iteration k_2, this inequality forces $z \geq z(x')$. As a result, CPA converges to an optimal solution in a finite number of steps.

Spread Network Inequalities .

For $\bar{x} \in X$, consider an optimal solution $\bar{y} = (\bar{y}^0, \ldots, \bar{y}^T)$, and its associated vector $\bar{\tau}$. Define $V_{\bar{x}}^t = \{i \in V : \bar{\tau}_i = t\},\ t \in \mathcal{T} \cup \{0\}$, and observe that $V_{\bar{x}} = \bigcup_{t=0}^{T} V_{\bar{x}}^t$. We denote by $G_{\bar{x}}(V_{\bar{x}}, A_{\bar{x}})$ an acyclic time-expanded network, called the *spread network*, with $T+1$ time levels. For node $i \in V_{\bar{x}}^t$ at time $t \in \mathcal{T}$, there exist at least Q nodes h such that $(h, i) \in A$ and $\bar{\tau}_h \leq t-1$. Let $S',\ |S'| = Q$, be any subset of such nodes. Then, for each $i \in V_{\bar{x}}^t,\ t \in \mathcal{T}$, add Q arcs $(h, i),\ \forall h \in S'$, to the spread network. Note that our construction implies that the spread network for $\bar{x}$ is not necessarily unique, because more than one such set S' may exist (see Figure 18 for an example).

For all $i \in V_{\bar{x}}$, define:

$$O_i = \{h \in V_{\bar{x}} :\ \text{there exists a directed path on } G_{\bar{x}} \text{ from node } h \text{ to node } i\},$$

and hence, $i \in O_i$. Next, consider any $\hat{x} \in X$ and define:

$$I_{\hat{x}} = \{i \in V_{\bar{x}} : O_i \cap P_{\hat{x}} \neq \emptyset\},$$

i.e., $I_{\hat{x}}$ is the set of all nodes $i \in V_{\bar{x}}$ that are either protected in $\hat{x}$, or such that there exists a directed path on $G_{\bar{x}}$ from some node $h \in P_{\hat{x}} \cap V_{\bar{x}}$ to node i. In the following lemma and theorem, we derive alternative valid inequalities for (104) by using the idea of the spread network.

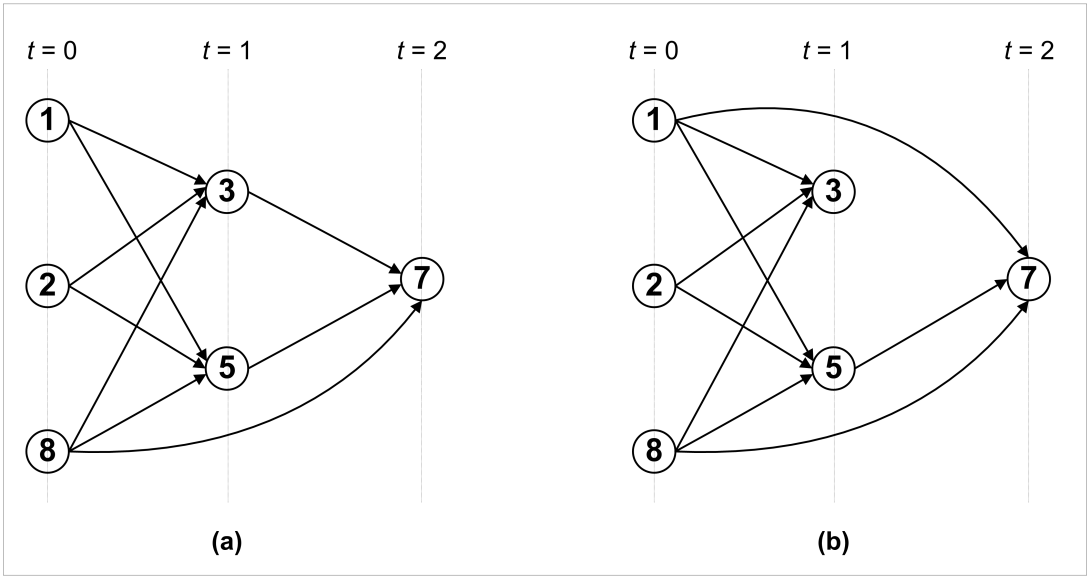


Figure 18: Two possible spread networks for Figure 17.

Lemma 15 *Given $\bar{x} \in X$, let $(\bar{y}^0, \ldots, \bar{y}^T)$ be optimal to ATT1($\bar{x}$) with corresponding vector $\bar{\tau}$. For any $\hat{x} \in X$, we have:*

$$z(\hat{x}) \geq \sum_{j \in V_{\bar{x}} \setminus I_{\hat{x}}} r_j^{\bar{\tau}_j}. \tag{111}$$

Theorem 17 *Let $(\bar{x}, \tilde{z})$ be an optimal solution to DEF-R, and suppose that $z(\bar{x}) > \tilde{z}$. Also, let $\mathcal{G}$ be any (undirected) acyclic graph that is constructed over $V_{\bar{x}}$, and denote by $\mathcal{A}$ its set of arcs. Finally, define $\mathcal{A}_j$, $\forall j \in V_{\bar{x}}$, as the set of arcs $(u, v) \in \mathcal{A}$ such that $u \in O_j$ and $v \in O_j$. Then, the following inequality:*

$$z \geq \sum_{j \in V_{\bar{x}}} r_j^{\bar{\tau}_j} \left(1 - \sum_{i \in O_j} x_i + \sum_{(u,v) \in \mathcal{A}_j} x_u x_v \right), \tag{112}$$

is valid to (104) *and cuts off $(\bar{x}, \tilde{z})$.*

Corollary 2 *Let $(\bar{x}, \tilde{z})$ be an optimal solution to DEF-R, and suppose that $z(\bar{x}) > \tilde{z}$. If $\mathcal{A} = \emptyset$ in Theorem 17, then the following inequality:*

$$z \geq z(\bar{x}) - \sum_{i \in V_{\bar{x}}} \left(\min \left\{ z(\bar{x}) - z_{\min}, \sum_{j \in V_{\bar{x}}:\ i \in O_j} r_j^{\bar{\tau}_j} \right\} \right) x_i, \tag{113}$$

is valid to (104) *and cuts off $(\bar{x}, \bar{z})$.*

Using Theorem 17 and Corollary 2, we can employ CPA equipped with valid inequalities of the form (112) or (113) instead of (110) in Step 3 of CPA. The motivation for using these inequalities stems from the following theorem that compares (110) to (113).

Theorem 18 *Inequality* (113) *is at least as strong as* (110)*.*

A similar theorem cannot be stated that compares the strength of inequalities (112) and (113). If inequality (113) was weakened by replacing the coefficients of x_i with $\sum_{j \in V_{\bar{x}}:\ i \in O_j} r_j^{\bar{\tau}_j}$ (instead of the minimum of that term and $z(\bar{x}) - z_{\min}$), then (112) would be at least as strong as (113) due to the subtraction of quadratic terms present in (112).

A further consideration in implementing CPA with valid inequalities (112) regards the linearization of the quadratic terms in these inequalities. By restricting the set of arcs that can belong to the set $\mathcal{A}$, over all generated inequalities (112), we can limit the number of quadratic terms that must be linearized. We linearize each quadratic term $x_i x_j$ by substituting it with a continuous variable $x_{ij}^L \geq 0$, and including the inequality $x_{ij}^L \geq x_i + x_j - 1$ in DEF-R. (The inequalities $x_{ij}^L \leq x_i$ and $x_{ij}^L \leq x_j$ usually required to linearize this quadratic term are not necessary, because optimization forces each x_{ij}^L-variable to take its smallest value allowed by x_i and x_j.)

Recall that multiple spread networks can be derived for a given $\bar{x} \in X$ and its optimal response $\bar{y} = (\bar{y}^0, \ldots, \bar{y}^T)$, each of which might correspond to a different valid inequality of the form (112) or (113). Given a candidate spread network, $G_{\bar{x}}$, corresponding to $\bar{x}$ and $\bar{y}$, we seek a mechanism for modifying $G_{\bar{x}}$ to an alternative spread network, $G'_{\bar{x}}$, such that the inequality (113) generated corresponding to $G'_{\bar{x}}$ is at least as strong as the one corresponding to $G_{\bar{x}}$.

Theorem 19 *Consider a spread network $G_{\bar{x}}(V_{\bar{x}}, A_{\bar{x}})$ for which there exist nodes $i, j, k \in V_{\bar{x}}$ such that $i \in V^-(k)$, $(i,k) \notin A_{\bar{x}}$, $(j,k) \in A_{\bar{x}}$, and a path exists from i to j on $G_{\bar{x}}$. Let $\widetilde{G}_{\bar{x}}$ be a modified spread network obtained by replacing arc (j,k) in $G_{\bar{x}}$ with arc (i,k). The valid inequality* (113) *induced by $\widetilde{G}_{\bar{x}}$ is at least as strong as that induced by $G_{\bar{x}}$.*

Figure 19 illustrates an instance of the problem with $T = 2$ and $Q = 3$ in which all rewards equal 1 and $z_{\min} = 1$. For a given $\bar{x}$, let $G_{\bar{x}}$ be a spread network presented in Figure 19a. Note that $z(\bar{x}) = 7$. Then, the following inequality

$$z \geq 7 - 3x_1 - 4x_2 - 4x_3 - 3x_4 - 2x_5 - 2x_6 - x_7,$$

is induced by $G_{\bar{x}}$ from Corollary 2. Next, suppose that there exists an arc $(4,7) \in A$ and note that $(4,6) \in A_{\bar{x}}$. Using Theorem 19, we obtain a modified spread network $\widetilde{G}_{\bar{x}}$ from $G_{\bar{x}}$ by adding arc $(4,7)$ and removing arc $(6,7)$. (See Figure 19b.) By using Corollary 2 for $\widetilde{G}_{\bar{x}}$, we obtain the inequality

$$z \geq 7 - 3x_1 - 4x_2 - 4x_3 - 3x_4 - 2x_5 - x_6 - x_7,$$

which is stronger than the inequality induced by $G_{\bar{x}}$ due to the x_6-coefficients in these inequalities.

It is worth noting that the inequality (112) induced by $\widetilde{G}_{\bar{x}}$ may not necessarily be as strong as that induced by $G_{\bar{x}}$ in general. Let $\mathcal{A} = \{(2,6), (3,6)\}$ be the set of arcs used to generate the quadratic terms in (112). Inequalities

$$z \geq 7 - 3x_1 - 4x_2 - 4x_3 - 3x_4 - 2x_5 - 2x_6 - x_7 + 2x_2x_6 + 2x_3x_6, \tag{114}$$

and

$$z \geq 7 - 3x_1 - 4x_2 - 4x_3 - 3x_4 - 2x_5 - x_6 - x_7 + x_2x_6 + x_3x_6, \tag{115}$$

are induced by $G_{\bar{x}}$ and $\widetilde{G}_{\bar{x}}$, respectively, from Theorem 17. To see that (114) and (115) do not dominate one another, we show that the RHS for one constraint need not always be larger than the RHS for the other constraint. For $x' = (0,0,0,0,0,1,0)$, note that the RHS of inequality (114) is 5, while the RHS for (115) is 6. However, for $x'' = (0,1,1,0,0,1,0)$, the RHS of (114) is 1, and the RHS of (115) is 0. ■

Algorithm 18 describes our method for modifying a given spread network using the idea of Theorem 19 in order to strengthen valid inequality (113).

Algorithm 18 examines all candidates for node k (as defined in Theorem 19) from among the nodes in $V_{\bar{x}}^T, \ldots, V_{\bar{x}}^2$, in that order. Given a choice of k, the algorithm starts by creating a list L_k, containing all nodes $j \in V^-(k)$ that are influenced at some time earlier than $\bar{\tau}_k$. The algorithm examines each arc $(j,k) \in A_{\bar{x}}$ in nonincreasing order of arc lengths (i.e., nodes j are considered in nonincreasing order of their $\bar{\tau}_j$-values). Algorithm 18 seeks a node $i \in L_k$, $(i,k) \notin A_{\bar{x}}$, having the smallest value of $\bar{\tau}_i$ such that there exists a path from node i to node j (i.e., such that $\texttt{ADJ}(i,j) = 1$). In case such node is found (with $\bar{\tau}_i < \bar{\tau}_j$), nodes i, j, and k satisfy the criteria of Theorem 19, and so we replace arc (j,k) with arc (i,k) in $A_{\bar{x}}$. This process repeats until no more arcs can be replaced.

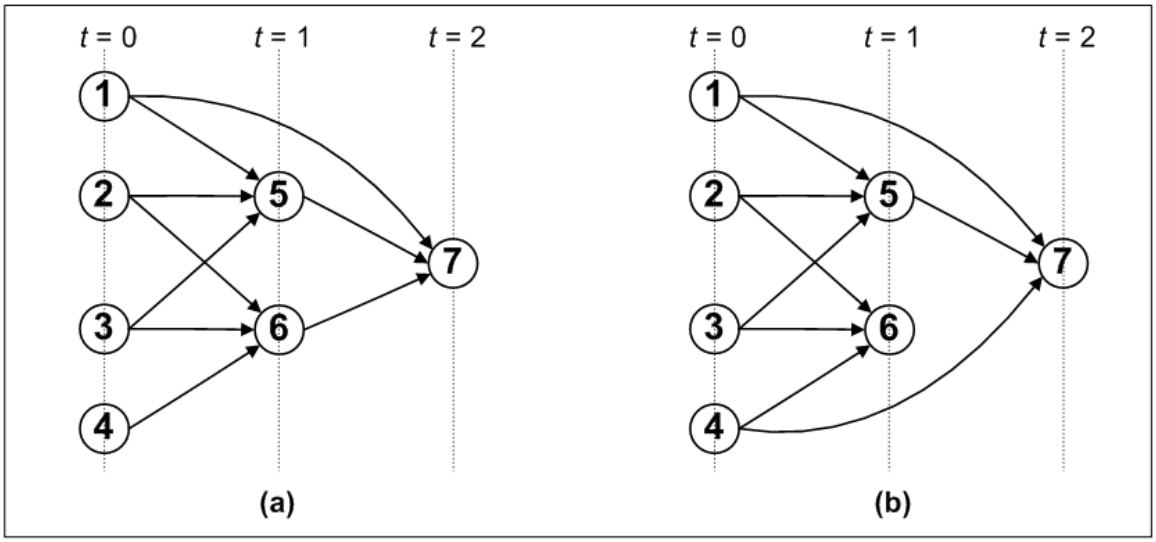


Figure 19: Spread network modification using Theorem 19.

Note that as Algorithm 18 proceeds, the spread network might be modified by "add" or "remove" operations performed in Step 15. However, the elements of matrix ADJ are never updated throughout the execution of Algorithm 18. This is due to the fact that $\texttt{ADJ}(i, j)$ correctly indicates the existence of a path between nodes i and j whenever it is examined in Step 14, even though the spread network might have been modified in earlier stages of Algorithm 18. To see this, suppose that at some stage of Algorithm 18, the value of $\texttt{ADJ}(i, j)$ is examined while visiting node $k \in V_{\bar{x}}$ in the for-loop at Step 4. Because this for-loop examines candidate nodes k in nonincreasing order of their $\bar{\tau}$-values, Algorithm 18 could have only modified the spread network by adding arcs (i', k') for some node $i' \in V_{\bar{x}}$ and $k' \in V_{\bar{x}}^t,\ t \geq \bar{\tau}_k$, or removing arcs (j', k') for some node $j' \in V_{\bar{x}}$ and $k' \in V_{\bar{x}}^t,\ t \geq \bar{\tau}_k$. Recall that the spread network contains no arcs (u, v) such that $\bar{\tau}_u \geq \bar{\tau}_v$. Therefore, the addition or deletion of arcs in Step 15 cannot create a new path, or disconnect an existing path, from node i to node j.

To analyze the complexity of Algorithm 18, observe that the construction of matrix ADJ takes $\mathcal{O}(Q|V_{\bar{x}}|^2)$ steps. For each node k examined in the for-loop at Steps 3 and 4, Algorithm 18 performs one sorting operation (Step 6), which is $\mathcal{O}(|V_{\bar{x}}| \log |V_{\bar{x}}|)$. For each node j examined in the for-loop in Step 7, Algorithm 18 executes $\mathcal{O}(|V_{\bar{x}}|)$ operations in the while-loop at Step 11 corresponding to each candidate node i. (Note that an arc (i, k) that is added to $A_{\bar{x}}$ after removing some arc (j, k) might be replaced later by some other arc (i', k) as the algorithm proceeds, which implies that a total of $\mathcal{O}(|V_{\bar{x}}|)$ nodes might be examined in the for-loop in Step 7). Therefore, for each node k examined in the for-loops at Steps 3 and 4, Algorithm 18 performs $\mathcal{O}(|V_{\bar{x}}|^2)$ operations, and hence, the overall complexity of Algorithm 18 is $\mathcal{O}(Q|V_{\bar{x}}|^2 + |V_{\bar{x}}|^3)$. In fact, the complexity can be more specifically stated as $\mathcal{O}(|V_{\bar{x}}|^3)$: If $Q \leq |V_{\bar{x}}|$, then this is obviously true, and if $Q > |V_{\bar{x}}|$, then every node in the spread network was influenced at time 0, $A_{\bar{x}}$ would necessarily be empty, and the algorithm would terminate in constant time.

Attacker's Problem Solution Approach. In order to generate the valid inequalities introduced before, we must solve the (NP-hard) attacker's problem. Therefore, the efficiency of our solution method is highly dependent on the time required to solve instances of the attacker's problem. This motivates further investigation of the attacker's problem with the aim of devising alternative formulations that can be more efficiently solved by mathematical optimization techniques.

Note that once the y^0-variables are fixed, the optimal value of each y_i^t-variable for $t \in \mathcal{T}$ can be readily determined via a polynomial-time procedure, which starts from time 1 and identifies the number of influenced nodes adjacent to node i at time 0. Then, $y_i^1 = 1$ if $x_i = 0$ and at least Q nodes adjacent to node i are influenced at time 0, and $y_i^1 = 0$ otherwise. By repeating the same operation for all other time periods, the optimal value of each y_i^t-variable will be either zero or one.

This observation suggests that a mathematical programming formulation for the attacker's problem that includes only $|V|$ binary variables may be attainable. However, the constraint (103f) cannot be relaxed in model (103), which indeed requires $\mathcal{O}(T|V|)$ binary variables. We investigate two alternative formulations for the attacker's problem that allow us to relax the binariness restriction on y_i^t-variables for $t \in \mathcal{T}$.

Exponential Model. For each $i \in V$, define $\mathcal{S}_i = \{S : S \subset V^-(i),\ |S| = |V^-(i)| - (Q-1)\}$. The following is a reformulation for model (103).

$$\textbf{ATT2}(\bar{x}) : z(\bar{x}) = \max \quad \sum_{i \in V} \left(r_i^0 y_i^0 + \sum_{t=1}^{T} r_i^t (y_i^t - y_i^{t-1}) \right) \tag{116a}$$

$$\text{s.t.} \quad y_i^t \leq y_i^0 + \sum_{j \in S} y_j^{t-1} \qquad \forall i \in V,\ t \in \mathcal{T},\ S \in \mathcal{S}_i \tag{116b}$$

$$y_i^t \in \{0,1\} \qquad \forall i \in V,\ t \in \mathcal{T} \tag{116c}$$

$$\text{Constraints (103b), (103d), and (103e).} \tag{116d}$$

Note that the only difference between models (103) and (116) lies in the constraints that govern the spread of influence. According to constraints (116b), if node $i \in V$ is initially influenced, then the RHS of constraints (116b) will be at least one for all $t \in \mathcal{T}$ and $S \in \mathcal{S}_i$, implying that node i will remain influenced at all time periods. Now, suppose that node $i \in V$ is not initially influenced, and examine constraints (116b) at time $t \in \mathcal{T}$. If fewer than Q nodes adjacent to node i are influenced at time $t-1$, then there exists a subset $\bar{S} \in \mathcal{S}_i$ such that no node in $\bar{S}$ is influenced at time period $t-1$. In this case, $y_i^t = 0$ due to constraint (116b) corresponding to $\bar{S}$. Otherwise, the RHS of constraints (116b) for all $S \in \mathcal{S}_i$ will be at least one for node i at time period t, and hence, $y_i^t = 1$ at optimality if $\bar{x}_i = 0$. The following theorem demonstrates that constraint (116c) can equivalently be relaxed to take continuous values.

Theorem 20 *Consider Problem ATT2($\bar{x}$) for any $\bar{x} \in X$, in which constraints* (116c) *are replaced with $0 \leq y_i^t \leq 1,\ \forall i \in V,\ t \in \mathcal{T}$. There exists an optimal solution $(\hat{y}^0, \ldots, \hat{y}^T)$ to this relaxation in which $\hat{y}_i^t \in \{0,1\},\ \forall i \in V,\ t \in \mathcal{T}$.*

Using Theorem 20, we henceforth relax constraint (116c) to $0 \leq y_i^t \leq 1,\ \forall i \in V,\ t \in \mathcal{T}$, in ATT2($\bar{x}$).

We now investigate the application of Benders' decomposition in solving model (116). Observe that model (116) reduces to the following linear program for given vectors $\bar{x}$ and $\bar{y}^0$:

$$\max \ \sum_{i\in V}\left(\sum_{t=1}^{T-1}(r_i^t - r_i^{t+1})y_i^t + r_i^T y_i^T\right) \tag{117a}$$

$$\text{s.t. } y_i^1 \le \bar{y}_i^0 + \sum_{j\in S}\bar{y}_j^0 \qquad \forall i\in V,\ S\in\mathcal{S}_i \tag{117b}$$

$$y_i^t - \sum_{j\in S} y_j^{t-1} \le \bar{y}_i^0 \qquad \forall i\in V,\ t\in\mathcal{T}\backslash\{1\},\ S\in\mathcal{S}_i \tag{117c}$$

$$y_i^t \le 1-\bar{x}_i \qquad \forall i\in V,\ t\in\mathcal{T} \tag{117d}$$

$$y_i^t \ge 0 \qquad \forall i\in V,\ t\in\mathcal{T}. \tag{117e}$$

Let $\pi_{i,S}^1$, $\pi_{i,S}^t$, and μ_i^t be the dual variables associated with (117b), (117c), and (117d), respectively. Defining $\mathcal{S}_{j,i} = \{S : S\in\mathcal{S}_j,\ i\in S\}$ as the set of all sets $S\in\mathcal{S}_j$ that include node i, we obtain the dual problem to (117) given $\bar{x}$ and $\bar{y}^0$:

$$\min \ \sum_{i\in V}\sum_{t=2}^{T}\sum_{S\in\mathcal{S}_i}\bar{y}_i^0\pi_{i,S}^t + \sum_{i\in V}\sum_{S\in\mathcal{S}_i}(\bar{y}_i^0 + \sum_{j\in S}\bar{y}_j^0)\pi_{i,S}^1 + \sum_{i\in V}\sum_{t=1}^{T}(1-\bar{x}_i)\mu_i^t \tag{118a}$$

$$\text{s.t. } \sum_{S\in\mathcal{S}_i}\pi_{i,S}^t - \sum_{j\in V^+(i)}\sum_{S\in\mathcal{S}_{j,i}}\pi_{j,S}^{t+1} + \mu_i^t \ge r_i^t - r_i^{t+1} \qquad \forall i\in V,\ t\in\mathcal{T}\backslash\{T\} \tag{118b}$$

$$\sum_{S\in\mathcal{S}_i}\pi_{i,S}^T + \mu_i^T \ge r_i^T \qquad \forall i\in V \tag{118c}$$

$$\pi_{i,S}^t \ge 0 \qquad \forall i\in V,\ S\in\mathcal{S}_i,\ t\in\mathcal{T} \tag{118d}$$

$$\mu_i^t \ge 0 \qquad \forall i\in V,\ t\in\mathcal{T}. \tag{118e}$$

Note that an optimal solution must exist to problem (118), because the objective function value is always nonnegative, and a feasible solution can be obtained by setting $\mu_i^t = r_i^t - r_i^{t+1},\ \forall i\in V,\ t\in\mathcal{T}$, with all π-variables equal to zero. Letting Ξ denote the set of all extreme points to problem (118), the Benders' master problem is given as:

$$\max \ \psi \tag{119a}$$

$$\text{s.t. } \psi \le \sum_{i\in V}(r_i^0 - r_i^1)y_i^0 + \sum_{i\in V}\sum_{S\in\mathcal{S}_i}\bar{\pi}_{i,S}^1(y_i^0 + \sum_{j\in S}y_j^0)$$

$$+\sum_{i\in V}\sum_{t=2}^{T}\sum_{S\in\mathcal{S}_i}\bar{\pi}_{i,S}^t y_i^0 + \sum_{i\in V}\sum_{t\in\mathcal{T}}(1-\bar{x}_i)\bar{\mu}_i^t \qquad \forall(\bar{\pi},\bar{\mu})\in\Xi \tag{119b}$$

$$y^0 \in Y(\bar{x}), \tag{119c}$$

with problem (117) being the Benders' subproblem. The restricted master problem (RMP) is given by (119) with only a limited set of dual extreme points, denoted by $\bar{\Xi}$, and corresponding constraints (119b).

Note, however, that problem (117) has an exponential number of constraints (117c). Therefore, we aim to solve this subproblem using methods other than linear programming techniques.

Let $\bar{y} = (\bar{y}^1, \ldots, \bar{y}^T)$ be an optimal solution to problem (117), and suppose $\bar{y}_i^t = 0$ for some unprotected node $i \in V$ and time $t \in \mathcal{T}$. Then, there must exist some $\bar{S}_i^t \in \mathcal{S}_i$ such that no node in $\bar{S}_i^t$ is influenced at time $t-1$. We refer to $\bar{S}_i^t$ as a *safe subset* for node $i \in V$ and time $t \in \mathcal{T}$. It is worth noting that a safe subset for node i such that $y_i^{t'} = 0$ is a safe subset for all time $t \leq t'$. We will thus discard the t-index from $\bar{S}_i^t$, and simply refer to $\bar{S}_i$ as a safe subset for node i for all time periods t such that $y_i^t = 0$. We provide a dual recovery algorithm (DRA) for identifying an optimal solution to problem (118) as follows.

Step 1 For all $i \in V$, if $\bar{x}_1 = 1$ or $y_i^T = 1$, then set $\bar{\mu}_i^T = r_i^T$ and $\bar{\pi}_{i,S}^T = 0,\ \forall S \in \mathcal{S}_i$. Otherwise, set $\bar{\mu}_i^T = 0$, $\bar{\pi}_{i,\bar{S}_i}^T = r_i^T$, and $\bar{\pi}_{i,S}^T = 0,\ \forall S \in \mathcal{S}_i \backslash \{\bar{S}_i\}$. Initialize $t = T$.

Step 2 If $t = 0$, then terminate. Otherwise, set $t = t - 1$ and proceed to Step 3.

Step 3 For every $i \in V$:

- **a.** If $\bar{x}_i = 1$, then $\bar{\mu}_i^t = r_i^t - r_i^{t+1} + \sum_{j \in V^+(i)} \sum_{S \in \mathcal{S}_{j,i}} \bar{\pi}_{j,S}^{t+1}$ and $\bar{\pi}_{i,S}^t = 0,\ \forall t \in \mathcal{T} \backslash \{T\},\ S \in \mathcal{S}_i$.
- **b.** If $\bar{x}_i = 0$ and $\bar{y}_i^t = 0$, then set $\bar{\mu}_i^t = 0$, $\bar{\pi}_{i,\bar{S}_i}^t = r_i^t - r_i^{t+1} + \sum_{j \in V^+(i)} \sum_{S \in \mathcal{S}_{j,i}} \bar{\pi}_{j,S}^{t+1}$, and $\bar{\pi}_{i,S}^t = 0,\ \forall S \in \mathcal{S}_i \backslash \{\bar{S}_i\}$.
- **c.** If $\bar{x}_i = 0$ and $\bar{y}_i^t = 1$, then set $\bar{\mu}_i^t = r_i^t - r_i^{t+1}$ and $\bar{\pi}_{i,S}^t = 0,\ \forall S \in \mathcal{S}_i$.

Proceed to Step 2.

In the following lemma, we establish the optimality of the solution identified by DRA.

Lemma 16 *Suppose that the solution $\bar{y} = (\bar{y}^1, \ldots, \bar{y}^T)$ is optimal to problem* (117)*. Let $\bar{S}_i$ be a safe subset for each unprotected node $i \in V$ such that $y_i^t = 0$ for some $t \in \mathcal{T}$. Then, DRA constructs an optimal solution $(\bar{\pi}, \bar{\mu})$ to problem* (118)*.*

An immediate result from Lemma 16 is that we only need to identify a single safe subset $\bar{S}_i$ for each unprotected node $i \in V$ that is not influenced at time 1. This allows us to discard S-indices from the π-variables when referring to problem (118), and to rewrite constraints (119b) as:

$$\psi \leq \sum_{i \in V} (r_i^0 - r_i^1) y_i^0 + \sum_{i \in V} \bar{\pi}_i^1 (y_i^0 + \sum_{j \in \bar{S}_i} y_j^0) + \sum_{i \in V} \sum_{t=2}^{T} \bar{\pi}_i^t y_i^0 + \sum_{i \in V} \sum_{t \in \mathcal{T}} (1 - \bar{x}_i) \bar{\mu}_i^t,$$

or equivalently,

$$\psi \leq \sum_{i \in V} \left(r_i^0 - r_i^1 + \bar{\pi}_i^1 + \sum_{j : i \in \bar{S}_j} \bar{\pi}_j^1 + \sum_{t=2}^{T} \bar{\pi}_i^t \right) y_i^0 + \sum_{i \in V} \sum_{t \in \mathcal{T}} (1 - \bar{x}_i) \bar{\mu}_i^t. \tag{120}$$

Inequality (120) can be strengthened by a standard coefficient tightening procedure as follows. Let η_i be the coefficient of variable $y_i^0,\ i \in V$, in (120). The following inequality:

$$\psi \leq \sum_{i \in V} \left(\min\{\eta_i,\ z_{\max}(\bar{x}) - \sum_{i \in V} \sum_{t \in \mathcal{T}} (1 - \bar{x}_i)\bar{\mu}_i^t\} \right) y_i^0 + \sum_{i \in V} \sum_{t \in \mathcal{T}} (1 - \bar{x}_i)\bar{\mu}_i^t, \tag{121}$$

is valid to (119), because $\eta_i \geq 0$ and $\bar{y}_i^0 \in \{0,1\},\ \forall i \in V$; $z_{\max}(\bar{x}) - \sum_{i \in V} \sum_{t \in \mathcal{T}} (1 - \bar{x}_i)\bar{\mu}_i^t \geq 0$ (noting that $\sum_{i \in V} \sum_{t \in \mathcal{T}} (1 - \bar{x}_i)\bar{\mu}_i^t$ is the optimal attacker's objective in the last inequality, which is no more than $z_{\max}(\bar{x})$); and $\psi \leq z_{\max}(\bar{x})$. Thus, when solving the Benders' master problem, we replace (119b) with (121).

We can also consider an alternative compact (polynomial-size) formulation for the attacker's problem, in which the binary restrictions on the y_i^t-variables can be relaxed. For each $i \in V$, arbitrarily order the nodes in $V^-(i)$ as $\{i_1, \ldots, i_{|V^-(i)|}\}$. Define $v_{imk}^t = 1$ if at least k of first m nodes in $V^-(i)$ are influenced at time $t-1$, and $v_{imk}^t = 0$ otherwise. By convention, we let $v_{imk}^t = 0,\ k > m$. Letting $\mathbb{N}_p = \{1, \ldots, p\}$ for any positive integer p, the following is a reformulation for model (103), given $\bar{x}$:

$$\textbf{ATT3}(\bar{x}) : \max \quad \sum_{i \in V} \left(r_i^0 y_i^0 \ + \sum_{t=1}^{T} r_i^t (y_i^t - y_i^{t-1}) \right) \tag{122a}$$

$$\text{s.t.} \quad v_{imk}^t \leq v_{i,m-1,k-1}^t \qquad \forall i \in V,\ t \in \mathcal{T},\ m \in \mathbb{N}_{|V^-(i)|},\ k \in \mathbb{N}_{\min\{m,Q\}} \tag{122b}$$

$$v_{imk}^t \leq v_{i,m-1,k}^t + y_{i_m}^{t-1} \quad \forall i \in V,\ t \in \mathcal{T},\ m \in \mathbb{N}_{|V^-(i)|},\ k \in \mathbb{N}_{\min\{m,Q\}} \tag{122c}$$

$$y_i^t \leq y_i^0 + v_{i,|V^-(i)|,Q}^t \qquad \forall i \in V,\ t \in \mathcal{T} \tag{122d}$$

$$y_i^t \leq 1 - \bar{x}_i \qquad \forall i \in V,\ t \in \mathcal{T} \tag{122e}$$

$$v_{imk}^t \in \{0,1\} \qquad \forall i \in V,\ t \in \mathcal{T},\ m \in \mathbb{N}_{|V^-(i)|},\ k \in \mathbb{N}_{\min\{m,Q\}} \tag{122f}$$

$$y_i^t \in \{0,1\} \qquad \forall i \in V,\ t \in \mathcal{T} \tag{122g}$$

$$y^0 \in Y(\bar{x}). \tag{122h}$$

The objective function (122a) is the same as the objective function in model (103). Constraints (122b) and (122c) enforce the definition of v_{imk}^t. To see this, suppose that fewer than k of the first m nodes in $V^-(i)$ are influenced at time $t-1$. If node i_m is influenced at time $t-1$, then at most $k-2$ of first $m-1$ nodes in $V^-(i)$ can be influenced at time $t-1$. This implies that $v_{i,m-1,k-1}^t = 0$, and constraints (122b) force $v_{imk}^t = 0$. Otherwise, if i_m is not influenced at time $t-1$, then at most $k-1$ of the first $m-1$ nodes in $V^-(i)$ are influenced at time $t-1$, i.e., $v_{i,m-1,k}^t = 0$, and constraints (122c) force $v_{imk}^t = 0$. On the other hand, if at least k of the first m nodes in $V^-(i)$ are influenced at time $t-1$, then at least $k-1$ of the first $m-1$ nodes were influenced at time $t-1$. Furthermore, either at least k of the first $m-1$ nodes were influenced, or node i_m itself was influenced at time $t-1$. Hence, $v_{i,m-1,k-1}^t = 1$ and $v_{i,m-1,k-1}^t + y_{i_m}^{t-1} \geq 1$, which allows $v_{imk}^t \geq 1$ (as will be the case at optimality). Constraints (122d) imply that node i cannot be influenced at time t unless either it has been initially influenced or at least Q of its adjacent nodes are influenced at time $t-1$. The binariness of the v- and y-variables and the budget limit are enforced by constraints (122f)–(122h). However, the following theorem demonstrates that constraints (122f) and (122g) can equivalently be relaxed to take continuous values.

Theorem 21 *Consider the relaxed version of ATT3($\bar{x}$) in which constraints* (122f) *and* (122g) *are replaced with the following constraints:*

$$0 \leq v_{imk}^t \leq 1 \qquad \forall i \in V,\ t \in \mathcal{T},\ m \in \mathbb{N}_{|V^-(i)|},\ k \in \mathbb{N}_{\min\{m,Q\}} \tag{123a}$$

$$0 \leq y_i^t \leq 1 \qquad \forall i \in V,\ t \in \mathcal{T}. \tag{123b}$$

There exists an optimal solution $(\hat{y}, \hat{v})$ to the relaxed problem in which $\hat{v}_{imk}^t \in \{0,1\},\ \forall i \in V,\ t \in \mathcal{T},\ m \in \mathbb{N}_{|V^-(i)|},\ k \in \mathbb{N}_{\min\{m,Q\}}$, and $\hat{y}_i^t \in \{0,1\},\ \forall i \in V,\ t \in \mathcal{T}$.

We thus replace (122f) and (122g) with (123a) and (123b), respectively. Note that while only $|V|$ binary variables are needed in model (122), the formulation requires the addition of $\mathcal{O}(TQ|V|^2)$ continuous variables. Table 5 compares the size of the proposed three formulations for the attacker's problem.

Computational Summary. Recall that ATT2 is designed to combat the growth of mathematical programming models as a function of T, wherein the DRA method executes a low-polynomial-time routine to calculate the impact of an attacker's action and generate a Benders' cut. The tradeoff is that ATT2 requires the solution of a (mixed-integer) master problem that may require the addition of many cuts. Let σ be the ratio of the number of nodes that cannot be initially attacked to $|V|$. We observe that when $\sigma = 40\%$, ATT1 still outperforms the other two variants. However, ATT2 outperforms ATT1 and ATT3 for $\sigma = 70\%$. Evidently, when σ changes from 40% to 70%, the Benders' master problem becomes less difficult to solve, and solving ATT2 becomes the most efficient approach.

For the defender's problem, recall that Step 2 of CPA requires the solution of the attacker's problem given a defender's decision vector. We employ formulation ATT1 to solve the attacker's problem, and find that CPA3 outperforms all other variants. In particular, CPA3 outperforms CPA1 as predicted by Theorem 18. Full results are given in Hemmati et al. (2014).

10 Dynamic Shortest Path Interdiction

Most contemporary network interdiction problems can be described as Stackelberg games in which two agents (a user and an attacker) with opposed interests interact in a network. In these games the user wishes to optimally transmit flows through the network, while the attacker optimally compromises some limited set of network elements to disturb the user operation. The shortest-path interdiction (SPI) problem is a two-stage game in which the attacker acts first, and the user acts second with full knowledge of the attacker's actions. Knowing that the user will select a shortest path between two known nodes, the attacker interdicts a subset of arcs by increasing their cost, or perhaps by preventing travel on those arcs. The SPI is typically formulated as a max-min optimization problem, in which the attacker's actions cannot change as the user travels the network.

In the SPI the user has the advantage of seeing the attacker's decisions before selecting its path. An alternative approach is taken in the robust shortest path (RSP) problem (Bertsimas and Sim,

2003), in which the user first selects (and commits to) a path, and then the attacker interdicts arcs to maximally increase the path's cost, given a budget constraint on the number of arcs that can be interdicted. In comparison to the SPI, the attacker is at an advantage in the RSP, having full knowledge of the user's path before taking any action.

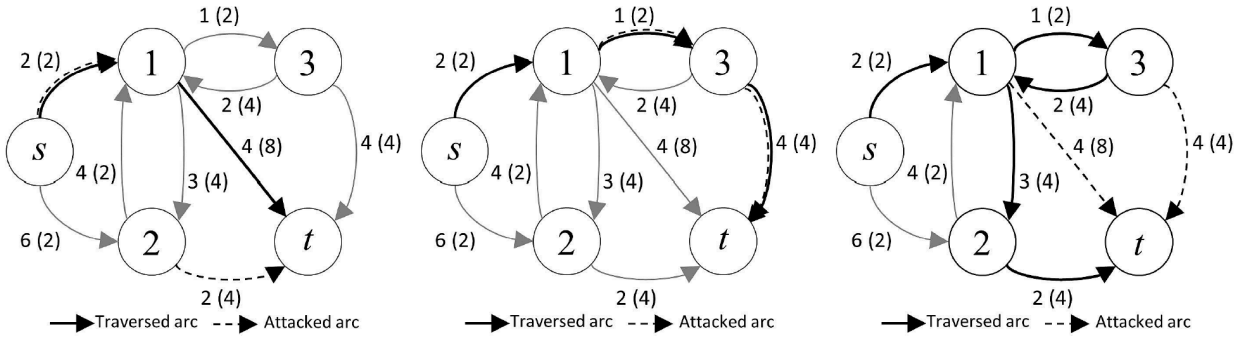
To illustrate the interaction between the user and attacker, consider the graph depicted in Figure 20. Uninterdicted traversal costs are shown alongside each arc. If an arc is interdicted, its cost increases by the value shown in parentheses. (For instance, traversing arc $(1, t)$ has a cost of 4 if the arc is not interdicted, and a cost of 12 if it is interdicted.) Suppose that the user wants to move from node s to node t and that the attacker can interdict two arcs. Figure 20a illustrates the optimal SPI solution. In this case the optimal strategy for the attacker is to interdict arcs $(s, 1)$ and $(2, t)$. Given these attacks, the user follows the path $s \to 1 \to t$, which has an optimal cost of $z^*_{SPI} = 8$. Figure 20b illustrates the optimal solution for the RSP, in which the optimal user's path is $s \to 1 \to 3 \to t$. Given this path, the attacker interdicts arcs $(1, 3)$ and $(3, t)$, producing an optimal objective of $z^*_{RSP} = 13$. Not surprisingly, $z^*_{RSP} \geq z^*_{SPI}$, because the user has the advantage of knowing the interdicted arcs in the SPI, whereas the attacker has the advantage of knowing the user's path in the RSP.

In this article we propose a *dynamic* shortest path interdiction (DSPI) problem, in which a cardinality-constrained attacker can interdict arcs whenever the user reaches a node in its path. The user sees all interdicted arcs, and is aware of the attacker's remaining budget. Accordingly, the user can adjust its path dynamically at any point in response to arc interdictions. Hence, the DSPI proceeds in multiple stages, where each stage consists of a (possibly empty) subset of arcs that are interdicted by the attacker, followed by an arc traversed by the user. These stages continue until the user reaches the destination node.

Figure 20c illustrates the DSPI. Initially, the attacker decides not to interdict any arc, and the user moves from s to 1. At this point, the attacker interdicts arc $(1, t)$, and the user moves from 1 to 3. Upon arrival at node 3, the attacker interdicts arc $(3, t)$ and the user decides to move back to 1. Because the attacker has no remaining budget, the user reaches the destination node by following the path $1 \to 2 \to t$. The optimal DSPI objective is thus given by $z^*_{DSPI} = 10$. This instance illustrates two insights into the nature of DSPI solutions. First, because the user knows the attacker's budget at all times, the user realizes upon visiting node 1 for the first time (and seeing the interdiction of $(1, t)$) that the attacker can interdict one more arc. Thus, using arc $(1, 2)$ at this point would induce the attacker to interdict arc $(2, t)$, trapping the user into traversing an expensive path to t. Second, note that unlike many other flow problems with nonnegative costs, it may be optimal for the user to return to previously visited nodes, as in this example.

The DSPI has received almost no attention in the literature, with the notable exception of a study by Khachiyan et al. (2008). This work examines a simplified version of the DSPI in which the attacker can remove up to a fixed number of outgoing arcs at each node. This simplified problem is proven to be polynomially solvable.

Problem Formulation. Consider a directed graph $G = (\mathcal{N}, \mathcal{A})$, where $\mathcal{N}$ is the set of nodes and $\mathcal{A}$ is the set of arcs. The user seeks to travel from source node $s \in \mathcal{N}$ to destination node



(a) z^*_{SPI}=8 (b) z^*_{RSP}=13 (c) z^*_{DSPI}=10

Figure 20: Shortest-path interdiction, robust shortest path, and dynamic shortest-path interdiction

$t \in \mathcal{N}$, $s \neq t$, at the minimum cost, considering that some arcs can be interdicted as the graph is traversed. In the context of this paper, the user's path might involve repeated traversals of some nodes. The cost of traversing arc $(i, j) \in \mathcal{A}$ is given by $c_{ij} \geq 0$, which is increased by $d_{ij} \geq 0$ units if interdicted. The attacker can interdict up to b arcs while the user traverses the graph. For a feasible interdiction set S (i.e., a subset of interdicted arcs with $|S| \leq b$), $\tilde{c}_{ij}(S)$ denotes the cost of traversing arc $(i, j) \in \mathcal{A}$. Formally,

$$\tilde{c}_{ij}(S) = \begin{cases} c_{ij} + d_{ij} & \text{if } (i,j) \in S, \\ c_{ij} & \text{if } (i,j) \notin S. \end{cases}$$

Let $\tilde{\mathbf{C}}(S)$ be a vector whose entries are given by $\tilde{c}_{ij}(S)$, for all $(i, j) \in \mathcal{A}$, and define $\mathcal{A}(i)$ as the set of arcs leaving node i, for each $i \in \mathcal{N}$. We assume without loss of generality that $\mathcal{A}(t) = \emptyset$. At every node in the user's path, the attacker must decide whether to expand the interdiction set S. As the following observation shows, the attacker needs to only consider interdicting those arcs in $\mathcal{A}(i)$.

Observation 1 *Suppose that the user is at node $i \in \mathcal{N} \setminus \{t\}$ in the current path. Then, it is optimal for the attacker to interdict only a subset of arcs in $\mathcal{A}(i)$.*

This observation holds because if it were optimal to interdict some arc $(j, k) \notin \mathcal{A}(i)$ while the user is at node i, then an optimal solution exists in which the attacker simply waits for the user to reach node j before interdicting (j, k). Revealing future attacks outside of $\mathcal{A}(i)$ allows the user to effectively avoid (or mitigate) the impact of the interdiction.

Consider a shortest-path tree to t, computed using the uninterdicted arc costs. The following lemma demonstrates that if the attacker uses its entire interdiction budget, it is optimal to interdict at least one arc in this shortest-path tree.

Lemma 17 *Let $\mathcal{T}$ be the set of arcs in some all-to-t shortest-path tree computed with costs $\tilde{\mathbf{C}}(\emptyset)$. If it is optimal for the attacker to interdict b arcs, then it is optimal for the attacker to interdict at least one arc in $\mathcal{T}$.*

Observation 2 *Lemma 17 requires the condition that the attacker's optimal solution exhausts the entire interdiction budget. The proof of this lemma utilizes the fact that the attacker exits the game after the last arc is interdicted, leaving the user to select a shortest path on the remaining network. If the optimal attack does not involve b interdictions, then Lemma 17 is no longer valid, as shown by the example depicted in Figure 21 where $b = 2$. The optimal strategy for the attacker is to interdict arc $(s, 2)$ when the user is at node s. The user then follows the path $s \to 2 \to t$ with no further interdictions: When the user reaches node 2, the attacker withholds action to prevent the user from traversing the cheaper route $2 \to 3 \to t$. As a result, the attacker does not interdict any arc in the unique shortest path tree $\mathcal{T} = \{(s, 1), (1, t), (2, 3), (3, t)\}$.*

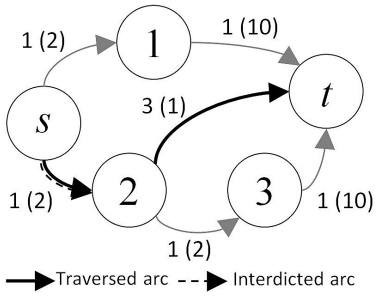


Figure 21: Optimal DSPI solution including no arcs from $\mathcal{T}$ when $b = 2$

To formally describe the attacker's optimization problem, define $z^*(S, i)$ as the optimal value of the game if the user is at node i and the attacker has already interdicted arcs in S. Intuitively, $z^*(S, i)$ is the cost of the optimal path from i to t, given that the attacker has interdicted arcs in S, and can still interdict $b - |S|$ additional arcs. Mathematically,

$$z^*(S, i) = \max_{\{S' | S' \subseteq \mathcal{A}(i) \setminus S,\, |S \cup S'| \leq b\}} \left\{ \min_{j \in FS(i)} \{z^*(S \cup S', j) + \tilde{c}_{ij}(S \cup S')\} \right\}, \tag{124}$$

where $FS(i) = \{j \in \mathcal{N} \mid (i, j) \in \mathcal{A}\}$. The attacker maximizes the value of the game in (124) by selecting additional arcs to interdict, denoted by S', so that $|S \cup S'| \leq b$ (note that $S' = \emptyset$ is allowed). Given the updated interdiction $S \cup S'$, the user then decides which node to visit next among those in $FS(i)$. To make this decision, the user examines each $j \in FS(i)$ and considers both the optimal value of the game at node j (given by $z^*(S \cup S', j)$) and the cost of traversing arc (i, j) (given by $\tilde{c}_{ij}(S \cup S')$). Note therefore that the optimal value of the game is given by $z^*(\emptyset, s)$, i.e., the optimal cost when the user is at node s and no arcs have yet been interdicted.

Define $\bar{z}(S, i)$ as the best achievable value of the game if the attacker *strictly* expands the interdiction set S while the user is at node i, i.e., if $|S'| > 0$ in (124). Mathematically,

$$\bar{z}(S, i) = \begin{cases} \max\limits_{\{S' | S' \subseteq \mathcal{A}(i) \setminus S,\, |S \cup S'| \leq b,\, S' \neq \emptyset\}} \left\{ \min\limits_{j \in FS(i)} \{z^*(S \cup S', j) + \tilde{c}_{ij}(S \cup S')\} \right\} & \text{if } i \neq t,\ |S| < b, \text{ and } S \neq \mathcal{A}(i), \\ 0 & \text{if } i = t, \text{ or } |S| = b, \text{ or } S = \mathcal{A}(i). \end{cases} \tag{125}$$

Define $\bar{\mathbf{z}}(S)$ as the $|\mathcal{N}|$-dimensional vector whose entries are given by $\bar{z}(S, i)$, for all $i \in \mathcal{N}$. Observe that if the user is at node t, or if the attacker has no remaining interdiction budget, or if

the attacker has previously interdicted all arcs in $\mathcal{A}(i)$, then we simply define $\bar{z}(S,i) = 0$. Using (125) we can rewrite (124) as

$$z^*(S,i) = \max\left\{\bar{z}(S,i), \min_{j \in FS(i)}\{z^*(S,j) + \tilde{c}_{ij}(S)\}\right\}. \tag{126}$$

Exact Algorithm for the DSPI. To solve the DSPI on a general network, we propose a dynamic-programming algorithm that calculates $z^*(S,i)$ for all $i \in \mathcal{N}$ and for all feasible interdiction sets S. Suppose that vector $\bar{\mathbf{z}}(S)$ is known, for some feasible interdiction set S. We first show how to calculate $z^*(S,i)$, for all $i \in \mathcal{N}$, using a modified version of Dijkstra's algorithm. Consider the algorithm FIS (which stands for "fixed interdiction set"), whose inputs are the graph G, the observed interdiction set S, and the vector $\bar{\mathbf{z}}(S)$. Algorithm FIS follows similar logic as used in Dijkstra's algorithm. However, in this case, we require the use of two labels per node. At node $i \in \mathcal{N}$, $\check{\ell}_i$ represents the user's shortest-path cost if the attacker takes no action when the user reaches node i. Thus, $\check{\ell}_i$ will equal the minimum value of $z^*(S,j) + \tilde{c}_{ij}(S)$, over all $j \in FS(i)$. Also, ℓ_i represents the node label from the attacker's perspective, and will thus be the maximum of $\check{\ell}_i$ and $\bar{z}(S,i)$. Note that when ℓ_i becomes permanently labeled (and is "correct"), then $\ell_i = z^*(S,i)$, allowing it to be employed in subsequent iterations of the algorithm.

We initially set $\check{\ell}_t = \ell_t = 0$ and $\check{\ell}_i = \ell_i = \infty$ for all $i \in \mathcal{N}\backslash\{t\}$. We use a priority queue, Q, to store the temporarily labeled nodes (initially $Q = \mathcal{N}$) and iteratively remove a node j having the smallest value of ℓ_j in Q. Once we remove a minimum-labeled node j from Q (declaring $\ell_j = z^*(S,j)$), we update $\check{\ell}_i$ for all nodes $i \in \mathcal{N}$ such that (i,j) belongs to $\mathcal{A}$. The label updates are done in the form $\check{\ell}_i = \min\{\check{\ell}_i, \ell_j + \tilde{c}_{ij}(S)\}$, as done in Dijkstra's algorithm, and $\ell_i = \max\{\check{\ell}_i, \bar{z}(S,i)\}$ by definition of ℓ_i. In particular, note that if $\bar{\mathbf{z}}(S) = \mathbf{0}$, then FIS becomes Dijkstra's algorithm.

The complexity of FIS is the same as Dijkstra's algorithm, because the computations in FIS only differ by one comparison in line 6 and one assignment in line 7. This complexity is given by $O(|\mathcal{N}|\log|\mathcal{N}| + |\mathcal{A}|)$ if a Fibonacci heap is used to handle the priority queue Q. The correctness of FIS is proven in Sefair and Smith (2015).

Consider the dynamic-programming algorithm DP-DSPI, whose inputs are the graph G and the attacker's budget b. Define $\Gamma(k)$ as the set of *all* cardinality-k subsets of $\mathcal{A}$, $\forall k = 0,\ldots,b$. Algorithm DP-DSPI initially calculates $z^*(S,i)$ for all cardinality-b interdiction sets that include at least one arc from $\mathcal{T}$ (see Lemma 17). Because the graph costs cannot be altered again by the attacker if b arcs have already been interdicted, this step can be done by solving an all-to-t shortest-path problem, for each $S \in \Gamma(b)$ such that $S \cap \mathcal{T} \neq \emptyset$. The algorithm then considers all possible sizes for an interdiction set (in decreasing order) from $b-1$ to 0. At iteration k ($0 \leq k \leq b-1$), $\bar{z}(S,i)$ is calculated for all cardinality-k sets in line 3. (Observe that at this step $z^*(\bar{S},i)$ has already been calculated for all $\bar{S} \in \Gamma(k')$, $\forall k' > k$.) Thus, $\bar{z}(S,i)$ can be calculated using (125). In line 4, $z^*(S,i)$ is calculated for all cardinality-k sets, which is possible because $\bar{z}(S,i)$ was calculated in line 3. The algorithm returns $z^*(\emptyset,s)$ upon termination in line 5.

Although exact, DP-DSPI(G,b) is exponential in running time and memory. Lines 1 and 4 of DP-DSPI(G, b) require the solution of $\sum_{k=0}^{b}\binom{|\mathcal{A}|}{k}$ total shortest-path problems. Line 3 requires the computation of (125) for every possible combination of interdiction set S, $|S| < b$, and node i.

Each such calculation requires $O(|FS(i)|)$ steps in the inner minimization loop and no more than $O(2^{|FS(i)|})$ iterations over the outer maximization loop, for a total of $O(|FS(i)|2^{|FS(i)|})$ computations. Because $|FS(i)| < |\mathcal{N}|$, the total number of computations required in line 3 of the algorithm is $O(\sum_{k=0}^{b-1} \binom{|\mathcal{A}|}{k} |\mathcal{N}|^2 2^{|\mathcal{N}|})$. Noting that $|\mathcal{N}|^2 2^{|\mathcal{N}|} > \max\{|\mathcal{N}| \log |\mathcal{N}|, |\mathcal{A}|\}$, the time complexity of this algorithm is given as:

$$O\left(\sum_{k=0}^{b-1} \binom{|\mathcal{A}|}{k} |\mathcal{N}|^2 2^{\mathcal{N}} + \binom{|\mathcal{A}|}{b} \left(|\mathcal{N}| \log |\mathcal{N}| + |\mathcal{A}|\right)\right).$$

The space complexity is governed by the required calculations in line 3 of the algorithm. For a given value of k, it is necessary at line 3 to refer to supersets $S \cup S'$ when computing $\bar{z}(S, i)$ for some $S \in \Gamma(k)$ and $i \in \mathcal{N}$. We therefore retain each computed value of z^* in our algorithm. (In fact, if k decreases beyond $b - |\mathcal{N}|$, then it would be permissible to "forget" z^*-values corresponding to larger interdiction sets. However, we would retain either those values, or at least the appropriate predecessor information, in order to recover the optimal combination of interdiction and user path at the end of the algorithm.) All other storage requirements (e.g., storing $\bar{z}$-values within DP-DSPI or ℓ-values within FIS) are dominated by the space needed to store the z^*-values. The space complexity required by this algorithm is therefore given by number of node-and-interdiction-set combinations:

$$O\left(\sum_{k=0}^{b} \binom{|\mathcal{A}|}{k} |\mathcal{N}|\right).$$

DSPI on Acyclic Graphs. The complexity of DP-DSPI(G,b) is exponential because its state space tracks the set of interdicted arcs. If no cycles are present in G, then only the *number* of arcs that have been interdicted must be recorded. This claim follows from the fact that all arcs that have been interdicted during the game emanate from nodes previously visited by the user, recalling Observation 1. These nodes cannot be revisited by the user because the graph is acyclic, and so the dynamic programming state space no longer needs to record which arcs have been interdicted.

Define $v^*(k, i)$ as the optimal value of the game if the attacker's remaining budget equals k and the user is at node i. We can determine this value as:

$$v^*(k,i) = \begin{cases} \max\limits_{\{S' | S' \subseteq \mathcal{A}(i),\, |S'| \leq \min\{k, |FS(i)|\}\}} \left\{ \min\limits_{j \in FS(i)} \left\{ v^*\left(k - |S'|, j\right) + \tilde{c}_{ij}(S') \right\} \right\} & \text{if } i \neq t, \\ 0 & \text{if } i = t. \end{cases} \tag{127}$$

In (127), the attacker maximizes the value of the game by selecting a subset S of arcs to interdict from $\mathcal{A}(i)$, given the remaining budget, k. Based on the set S', the user then decides which node $j \in FS(i)$ to visit next, considering the value of the game at j (given by $v^*(k - |S'|, j)$) and the cost of traversing arc (i, j) (given by $\tilde{c}_{ij}(S')$).

To calculate $v^*(k, i)$, we begin by examining the optimal user strategy, given that the attacker decides to interdict arcs $S' \subseteq \mathcal{A}(i)$. Let $m = |S'|$, and assume that $m \leq k$. If the user selects an

arc (i, j) that does not belong to S', then the user visits node j^U, determined by

$$j^U \in \operatorname*{argmin}_{j:(i,j)\in\mathcal{A}(i)\setminus S'} \{v^*(k-m, j) + c_{ij}\},$$

where j^U is not defined if $S' = \mathcal{A}(i)$. If the user selects an arc $(i, j) \in S'$, then the user would visit a node j^I such that

$$j^I \in \operatorname*{argmin}_{j:(i,j)\in S'} \{v^*(k-m, j) + c_{ij} + d_{ij}\}.$$

Again, j^I is undefined if $S' = \emptyset$. Define $v^U(k, i, S') = v^*(k-m, j^U) + c_{ij^U}$ if j^U exists, and $v^U(k, i, S') = \infty$ otherwise. Likewise, define $v^I(k, i, S') = v^*(k-m, j^I) + c_{ij^I} + d_{ij^I}$ if j^I exists, and $v^I(k, i, S') = \infty$ otherwise. We then have that $v^*(k, i, S') = \min\{v^U(k, i, S'), v^I(k, i, S')\}$ is the value of the game if the attacker interdicts arcs in S' when the user is at node i and k more arcs can be interdicted.

We now turn our attention to the attacker's problem of maximizing $v^*(k, i, S')$ over all sets S' : $|S'| = m$. Suppose that the values of $v^*(k-m, j) + c_{ij}$ are sorted in nondecreasing order for $j = j_1^{k-m}, \ldots, j_{|FS(i)|}^{k-m}$, such that $v^*(k-m, j_1^{k-m}) + c_{ij_1^{k-m}} \le \cdots \le v^*(k-m, j_{|FS(i)|}^{k-m}) + c_{ij_{|FS(i)|}^{k-m}}$. The following lemma states an optimal interdiction strategy.

Lemma 18 *Consider a DAG $G = (\mathcal{N}, \mathcal{A})$, a node $i \in \mathcal{N}$, and a positive integer k. Suppose that if the user reaches node i and the attacker has a remaining interdiction budget of k arcs, the attacker chooses to interdict m arcs, where $0 \le m \le \min\{k, |FS(i)|\}$. Then an optimal interdiction solution exists in which arcs (i, j_p^{k-m}) are interdicted, for $p = 1, \ldots, m$.*

We now use Lemma 18 to derive a polynomial-time dynamic-programming algorithm, which we call DP-DSPI-DAG. Algorithm DP-DSPI-DAG calculates $v^*(k, i)$ by processing nodes in $\mathcal{N} \setminus \{t\}$ in reverse topological order. That is, if $(i, j) \in \mathcal{A}$, then DP-DSPI-DAG processes j before i. (Recall that a topological ordering always exists when G is acyclic.) This strategy ensures that when calculating $v^*(k, i)$ using (127), for some $i \in \mathcal{N} \setminus \{t\}$ and $k = 0, \ldots, b$, all values $v^*(k', j)$, $\forall j \in FS(i)$ and $k' = 0, \ldots, b$, have been precomputed. For each node $i \in \mathcal{N} \setminus \{t\}$, algorithm DP-DSPI-DAG uses the labels $\check{v}(k-m, i)$ to represent the user's shortest-path cost from i to t if the attacker's remaining budget is $k - m$ (after m arcs in $\mathcal{A}(i)$ are interdicted) and the user leaves i using one of the interdicted arcs.

Algorithm DP-DSPI-DAG first sets $v^*(k, t) = 0$, $\forall k = 0, \ldots, b$. Then, the algorithm selects the next node $i \in \mathcal{N} \setminus \{t\}$ to examine, and for each possible budget $k = 0, \ldots, b$, computes the value for $v^*(k, i)$ in lines 4–12. For node i and budget k, we initialize $v^*(k, i) = 0$ and $\check{v}(k, i) = \infty$ (because no arcs in $\mathcal{A}(i)$ have yet been interdicted). In line 5, the algorithm sorts the values of $v^*(k, j) + c_{ij}$, $\forall j \in FS(i)$, in nondecreasing order. This sorting is used in both the current and in future iterations to reduce computational effort in determining optimal attacks according to Lemma 18. The for-loop in line 6 calculates the attacker's best strategy by varying the number, m, of interdicted arcs, from 0 to $\min\{k, |FS(i)|\}$. If $m > 0$ then $\check{v}(k-m, i)$ is updated in lines 7–8 in the form $\check{v}(k-m, i) = \min\{\check{v}(k-m, i), v^*(k-m, j_m^{k-m}) + c_{ij_m^{k-m}} + d_{ij_m^{k-m}}\}$. Prior to executing line 8, $\check{v}(k-m, i)$ represents the minimum value of $v^*(k-m, j) + c_{ij} + d_{ij}$ over the $m-1$ interdicted arcs $(i, j) : j \in \{j_1^{k-m}, \ldots, j_{m-1}^{k-m}\}$. Thus, the second if-condition

in line 7 seeks to update v^i_{k-m} by examining the updated traversal cost of arc (i, j^{k-m}_m), i.e., the next arc to be interdicted in the sorted list. The algorithm updates $v^*(k,i)$ in lines 9–12 in the form $v^*(k,i) = \max\{v^*(k,i), \min\{\check{v}(k-m,i), v^*(k-m, j_{m+1}) + c_{ij_{m+1}}\}\}$, if $m < |FS(i)|$, or $v^*(k,i) = \max\{v^*(k,i), \check{v}(k-m,i)\}$, if $m = |FS(i)|$. The algorithm returns $v^*(b,s)$ as the optimal objective upon termination in line 13.

To calculate the time complexity of DP-DSPI-DAG, note that the topological order required in line 2 can be obtained in $O(|\mathcal{N}| + |\mathcal{A}|)$ time by using a depth-first-search (DFS) algorithm. Moreover, the sorting procedure required in line 5 can be done in $O(|FS(i)| \log |FS(i)|)$ time. (If b is small relative to $|FS(i)|$, then we could instead perform a partial sorting to find the $(b+1)$-smallest values of $v^*(k-m,j) + c_{ij}, \forall j \in FS(i)$. This method could be performed in $O(|FS(i)| + b \log b)$ time.) The overall complexity is given by

$$
\begin{aligned}
& O\left(|\mathcal{N}| + |\mathcal{A}| + \sum_{i \in \mathcal{N}\setminus\{t\}} \sum_{k=0}^{b} (|FS(i)| \log |FS(i)| + \min\{k, |FS(i)|\}) \right) \\
\leq\ & O\left(|\mathcal{N}| + |\mathcal{A}| + (b+1) \sum_{i \in \mathcal{N}\setminus\{t\}} |FS(i)| \log |FS(i)| + \sum_{i \in \mathcal{N}\setminus\{t\}} \sum_{k=0}^{b} |FS(i)| \right) \\
\leq\ & O\left(|\mathcal{N}| + |\mathcal{A}| + (b+1)|\mathcal{A}| \log |\mathcal{N}| + (b+1)|\mathcal{A}| \right) \\
\leq\ & O(|\mathcal{A}|^2 \log |\mathcal{N}|).
\end{aligned}
$$

The first inequality holds given that $\min\{k, |FS(i)|\} \leq |FS(i)|$. The second inequality is obtained by observing that $\sum_{i \in \mathcal{N}\setminus\{t\}} |FS(i)| \log |FS(i)| \leq \log |\mathcal{N}| \sum_{i \in \mathcal{N}\setminus\{t\}} |FS(i)|$ and $\sum_{i \in \mathcal{N}\setminus\{t\}} |FS(i)| = |\mathcal{A}|$. Because $b \leq |\mathcal{A}|$, the algorithm's complexity is $O(|\mathcal{A}|^2 \log |\mathcal{N}|)$. In terms of space, the bottleneck is the storage of the v^*-values, which results in $O(|\mathcal{N}||\mathcal{A}|)$ storage.

Lower and Upper Bounds. First, suppose that we select a subset of arcs, $\hat{S}$, that the attacker must interdict before the game begins, thus allowing the attacker to interdict only $k - |\hat{S}|$ more arcs while the user traverses a path from s to t. We denote this game DSPI($\hat{S}$), and define $z_{LB}(\hat{S})$ as the optimal value of this game. Note that $z_{LB}(\hat{S}) \leq z_{LB}(\overline{S})$ holds true for any $\hat{S} \supseteq \overline{S}$. This inequality is due to the fact that the attacker's problem in DSPI($\hat{S}$) is a restriction of that in DSPI($\overline{S}$), and moreover, the user has more information in DSPI($\hat{S}$) as to the attacker's planned interdictions than in DSPI($\overline{S}$).

Hence, consider *a priori* sets $\hat{S}^0, \ldots, \hat{S}^b$, where $\hat{S}^0 = \emptyset$, and for $m = 1, \ldots, b$, we have $\hat{S}^m \supset \hat{S}^{m-1}$ and $|\hat{S}^m| = |\hat{S}^{m-1}| + 1$, i.e., $\hat{S}^m$ has one arc in addition to those in $\hat{S}^{m-1}$. Clearly, $z_{LB}(\hat{S}^0) = z^*_{DSPI}$, since this corresponds to the case in which no arcs are interdicted in an *a priori* phase, just as in the DSPI. At the other end of the spectrum, $z_{LB}(\hat{S}^b)$ corresponds to the situation in which the attacker exhausts the entire interdiction budget before the user acts. Hence, $z_{LB}(\hat{S}^b) \leq z^*_{SPI}$, with equality holding only if $\hat{S}^b$ optimizes SPI. This analysis yields the bound hierarchy

$$
z_{LB}(\hat{S}^b) \leq z_{LB}(\hat{S}^{b-1}) \leq \cdots \leq z_{LB}(\hat{S}^1) \leq z_{LB}(\hat{S}^0) = z^*_{DSPI}.
$$

Also, if S^*_{SPI} optimizes the SPI and $\hat{S}^m \subseteq S^*_{SPI}$ for some $m \in \{0, \ldots, b\}$, then $z^*_{SPI} \leq z_{LB}(\hat{S}^m) \leq z^*_{DSPI}$.

Consider now the algorithm DSPI-LB, whose inputs are the graph, G, the attacker's budget, b, and the number of arcs to interdict in advance, Δ. Algorithm DSPI-LB initially selects any set of arcs $\hat{S}$ of size Δ. Then, the costs of arcs in $\hat{S}$ are revised to reflect the fact that those arcs are already interdicted. The algorithm finishes by solving the DSPI with the updated budget and arc costs, obtaining $z_{LB}(\hat{S})$.

Although DSPI-LB produces a valid lower bound for any $\hat{S}$, our implementation strategy chooses a set $\hat{S} \subseteq S^*_{SPI}$ to guarantee bounds that are at least as tight as z^*_{SPI}. After solving the SPI to identify S^*_{SPI}, we construct $\hat{S}$ by choosing the Δ arcs in S^*_{SPI} that are as close to s as possible (as measured by the shortest unweighted path from node s to an arc's from-node). In doing so, the attacker reveals as little information as possible regarding future interdiction plans, which then tends to yield a tighter lower bound.

As an alternative of DSPI-LB, consider problem DSPI-EXP (where EXP stands for "expired attacks"). Problem DSPI-EXP essentially modifies the game as follows. When the user is at some node $i \in \mathcal{N}$, the cost of an interdicted arc $(i, j) \in \mathcal{A}(i)$ becomes $c_{ij} + d_{ij}$ during the user's next move, as in the DSPI. However, after the user traverses some arc in $\mathcal{A}(i)$, the cost of arc (i, j) reverts to c_{ij}, i.e., the interdiction expires after the user's next arc traversal. Define z^*_{EXP} as the optimal value of DSPI-EXP. The motivation behind studying DSPI-EXP is twofold. One, because interdictions last for only one arc traversal instead of over the remainder of the game, we have that $z^*_{EXP} \leq z^*_{DSPI}$. Two, because interdictions expire, then just as in the case in which G is a DAG, the state space for a dynamic program needs to record only the remaining number of attacks rather than the set of previously attacked arcs.

We now describe an algorithm, DP-DSPI-EXP, for solving this modified game. As in DP-DSPI-DAG, we define $v^*(k, i)$ as the optimal value of the game, given that the user is at node i and the attacker has a remaining budget of k. Moreover, define u_{kim} as the optimal value of the game if the user is at node i and the attacker, having a budget of k, decides to interdict m arcs in $\mathcal{A}(i)$. If $m > |\mathcal{A}(i)|$, then u_{kim} is not defined. Algorithm DP-DSPI-EXP also uses $\check{v}(k, i)$, the user's shortest-path cost from i to t if the attacker's remaining budget is k after arcs in $\mathcal{A}(i)$ have been interdicted, and the user leaves i using one of the interdicted arcs.

The algorithm proceeds in $b + 1$ stages, computing all values of $v^*(k, i)$ in stage k, for each $i \in \mathcal{N} \setminus \{t\}$ and $k = 0, \ldots, b$. Stage 0 is performed by executing Dijkstra's algorithm, because no interdictions can occur when $k = 0$. At every subsequent stage, we execute two phases: Phase one, which consists of lines 4–15, and phase two, which consists of lines 16–23.

Phase one calculates u_{kim}, for each $i \in \mathcal{N} \setminus \{t\}$ and $m = 1, \ldots, \min\{k, |FS(i)|\}$, using previously computed values of $v^*(k, j)$, for $j \in FS(i)$ and $k = 0, \ldots, b$. The for-loop in line 4 iterates over all non-terminal nodes in $\mathcal{N}$, while line 5 sorts the $v^*(k - 1, j) + c_{ij}$ values as done in DP-DSPI-DAG. After initializing v^*- and $\check{v}$-values, the for-loop starting on line 7 iterates over every possible number m of interdictions that the attacker could choose at node i with a budget of k. Lines 8 and 9 update $\check{v}$, while lines 10–13 calculate u_{kim}, and lines 14 and 15 update v^* as appropriate. Hence, $v^*(k, i)$ equals $\max_{m \in \{1, \ldots, \min\{k, |FS(i)|\}\}} \{u_{kim}\}$ after the inner for-loop (indexed by m) is complete.

In phase two, we execute a modified Dijkstra's algorithm similar to FIS in order to obtain the remaining u_{ki0}-values, and to finalize $v^*(k,i)$, $\forall i \in \mathcal{N}$. In this case, node labels are given by $\max\{v^*(k,i), u_{ki0}\}$, $\forall i \in \mathcal{N}$, where $v^*(k,i)$ is initially computed in phase one and u-values are initialized as $u_{ki0} = \infty$, $\forall i \in \mathcal{N} \setminus \{t\}$, and $u_{kt0} = 0$. The label updates are done in the form $u_{ki0} = \min\{u_{ki0}, v^*(k,j) + c_{ij}\}$ and $v^*(k,j) = \max\{v^*(k,j), u_{kj0}\}$.

The time complexity of DP-DSPI-EXP is driven by the steps in lines 2 to 15, which is essentially the same complexity as DP-DSPI-DAG. The complexity of phase two is given by $O(|\mathcal{N}| \log |\mathcal{N}| + |\mathcal{A}|)$, which is the same as in FIS. Because $|\mathcal{A}|^2 \log |\mathcal{N}| \geq |\mathcal{N}| \log |\mathcal{N}| + |\mathcal{A}|$, the overall complexity of DP-DSPI-EXP is given by $O(|\mathcal{A}|^2 \log |\mathcal{N}|)$. In terms of space, the bottleneck is the storage of the sorted values in line 5, which results in $O(|\mathcal{N}|^2|\mathcal{A}|)$ storage.

Computing an upper bound on z^*_{DSPI} is performed by restricting the user rather than the attacker. This restriction is performed by simply removing arcs from G to form a graph $G' = (\mathcal{N}, \mathcal{A}')$, with $\mathcal{A}' \subseteq \mathcal{A}$. By choosing $\mathcal{A}'$ so that G' is a DAG, we can then solve the DSPI to optimality in polynomial time using DP-DSPI-DAG over G'. Letting z_{UB} equal the optimal DSPI objective over G', we then have that $z^\star_{DSPI} \leq z_{UB}$.

To choose G', we select a simple path from node s to node t, which we denote by $\mathcal{P}$. We then eliminate arcs other than those in $\mathcal{P}$ until the resulting graph is acyclic. By doing so, we guarantee that G' contains at least one path from s to t. Algorithm DSPI-UB formally states the upper bounding method.

The transformation of G into a DAG in line 2 can be done in $O(|\mathcal{N}| + |\mathcal{A}|)$ time by using a modification of a DFS algorithm in which all arcs in $\mathcal{P}$ are visited first, and then all arcs $(i,j) \in \mathcal{A}$ that connect a node i to an ancestor j in the DFS tree (i.e., "back arcs") are eliminated.

Note that by randomizing either the generation of $\mathcal{P}$ and/or the way in which G is transformed to G', we could obtain several upper bounds by solving DP-DSPI-DAG over alternative acyclic subgraphs of G, and retain the best (lowest) bound. A natural question then arises as to whether or not there exists an "ideal" subgraph G' that yields a tight upper bound of z^*_{DSPI}. The answer to this question is no, as evidenced by the example depicted in Figure 20c. In that example the unique optimal solution requires the user to visit node 1 twice, which would not be allowed in any DAG. Also, while we may intuitively seek a subgraph having as many arcs in G as possible, the problem of eliminating the fewest arcs in G so that G' becomes acyclic is equivalent to the *minimum feedback arc set* problem, which is NP-hard.

All computational results of our procedures and proofs can be found in Sefair and Smith (2015). Additionally, the same authors recently submitted another portion of this work on the *Dynamic Assignment Interdiction* problem, which was initiated due to DTRA support.

Algorithm 18: Revising an existing spread network using Theorem 19

Let $G_{\bar{x}} = (V_{\bar{x}}, A_{\bar{x}})$ be a spread network with corresponding vector $\bar{\tau}$;
Define ADJ as a $|V_{\bar{x}}| \times |V_{\bar{x}}|$ matrix, where $\texttt{ADJ}(i,j) = 1$ if there exists a path from node i to node j on $G_{\bar{x}}$, and $\texttt{ADJ}(i,j) = 0$ otherwise;
for $t = 0$ *to* $T-2$ **do**
  for *all nodes* $k \in V_{\bar{x}}^{T-t}$ **do**
    Initialize L_k as an array of all nodes $j \in V^-(k)$ such that $\bar{\tau}_j < T - t$;
    Sort nodes in L_k based on nonincreasing values of $\bar{\tau}$;
    for $p = 1$ *to* $|L_k|$ **do**
      Let j be the pth node in L_k;
      if $(j,k) \in A_{\bar{x}}$ **then**
        Set $q = |L_k|$, and let i be the qth node in L_k;
        Set flag `repeat = true`;
        while *(`repeat == true`)* **do**
          if $(i,k) \notin A_{\bar{x}}$ **then**
            if $\texttt{ADJ}(i,j) = 1$ **then**
              Remove arc (j,k) from $A_{\bar{x}}$ and add arc (i,k) to $A_{\bar{x}}$;
              Set `repeat = false`;
            end
            else
              Set $q = q - 1$, and let i be the qth node in L_k;
              if $\bar{\tau}_i \geq \bar{\tau}_j$ **then**
                Set `repeat = false`;
              end
            end
          end
        end
      end
    end
  end
end

Table 5: Size comparison of attacker's problem formulations.

Model	Binary Variables	Continuous Variables	Constraints
ATT1	$\mathcal{O}(T\|V\|)$	0	$\mathcal{O}(T\|V\|)$
ATT2	$\mathcal{O}(\|V\|)$	$\mathcal{O}(T\|V\|)$	$\mathcal{O}(T\|V\|\binom{\|V\|}{Q})$
ATT3	$\mathcal{O}(\|V\|)$	$\mathcal{O}(TQ\|V\|^2)$	$\mathcal{O}(TQ\|V\|^2)$

Algorithm 19: FIS(G, S, $\bar{\mathbf{z}}(S)$)

Initialize $\ell_i = \check{\ell}_i = \infty$, for each $i \in \mathcal{N} \setminus \{t\}$, and $\ell_t = \check{\ell}_t = 0$;
Let Q be the set of temporarily labeled nodes, with $Q = \mathcal{N}$ initially;
for $h = 1, \ldots, |\mathcal{N}|$ **do**
    Let j be a node having the minimum (temporary) label ℓ_j in Q, and remove j from Q;
    for *nodes* $i : (i, j) \in \mathcal{A}$ **do**
        if $\check{\ell}_i > \ell_j + \tilde{c}_{ij}(S)$ *and* $\check{\ell}_i > \bar{z}(S, i)$ **then**
            Set $\check{\ell}_i = \ell_j + \tilde{c}_{ij}(S)$ and $\ell_i = \max\{\check{\ell}_i, \bar{z}(S, i)\}$;
        end
    end
end

Algorithm 20: DP-DSPI(G, b)

Calculate $z^*(S, i)$ for each $S \in \Gamma(b)$, such that $S \cap \mathcal{T} \neq \emptyset$, and $i \in \mathcal{N}$ via FIS(G, S, $\mathbf{0}$);
for $k = b - 1$ *down to* $k = 0$ **do**
    Calculate $\bar{z}(S, i)$, $\forall S \in \Gamma(k)$ and $\forall i \in \mathcal{N}$, via the recursion given by (125);
    Calculate $z^*(S, i)$, $\forall S \in \Gamma(k)$ and $\forall i \in \mathcal{N}$, via FIS($G$, S, $\bar{\mathbf{z}}(S)$);
end
Return $z^*(\emptyset, s)$;

Algorithm 21: DP-DSPI-DAG(G, b)

Initialize $v^*(k, t) = 0$, $\forall k = 0, \ldots, b$;
for $i \in \mathcal{N} \setminus \{t\}$ *taken in reverse topological order* **do**
    for $k = 0, \ldots, b$ **do**
        Initialize $v^*(k, i) = -\infty$ and $\check{v}(k, i) = \infty$;
        Sort $v^*(k, j) + c_{ij}$, $\forall j \in FS(i)$, such that
        $v^*(k, j_1^k) + c_{ij_1^k} \leq \cdots \leq v^*(k, j_{|FS(i)|}^k) + c_{ij_{|FS(i)|}^k}$;
        for $m = 0, \ldots, \min\{k, |FS(i)|\}$ **do**
            if $m > 0$ ***and*** $\check{v}(k - m, i) > v^*(k - m, j_m^{k-m}) + c_{ij_m^{k-m}} + d_{ij_m^{k-m}}$ **then**
                $\check{v}(k - m, i) = v^*(k - m, j_m^{k-m}) + c_{ij_m^{k-m}} + d_{ij_m^{k-m}}$;
            end
            if $m < |FS(i)|$ ***and*** $v^*(k, i) < \min\{\check{v}(k - m, i), v^*(k - m, j_{m+1}^{k-m}) + c_{ij_{m+1}^{k-m}}\}$ **then**
                $v^*(k, i) = \min\{\check{v}(k - m, i), v^*(k - m, j_{m+1}^{k-m}) + c_{ij_{m+1}^{k-m}}\}$;
            end
            if $m = |FS(i)|$ *and* $v^*(k, i) < \check{v}(k - m, i)$ **then**
                $v^*(k, i) = \check{v}(k - m, i)$;
            end
        end
    end
end
Return $v^*(b, s)$;

Algorithm 22: DSPI-LB(G, b, Δ)

1 Select a set of arcs $\hat{S} \subseteq \mathcal{A}$ such that $|\hat{S}| = \Delta$;

2 For all $(i, j) \in \hat{S}$, revise c_{ij} to equal $c_{ij} + d_{ij}$, and revise $d_{ij} = 0$;

3 Compute $z_{LB}(\hat{S})$ by solving the DSPI with attacker's budget $b - \Delta$ and revised arc costs computed above;

Algorithm 23: DP-DSPI-EXP(G, b)

Compute $v^*(0,i)$, $\forall i \in \mathcal{N}$, via Dijkstra's algorithm using costs c_{ij}, $\forall (i,j) \in \mathcal{A}$;
for $k = 1, \ldots, b$ **do**
  Set $v^*(k,t) = 0$;
  for $i \in \mathcal{N} \setminus \{t\}$ **do**
    Sort $v^*(k-1,j) + c_{ij}$, $\forall j \in FS(i)$, such that
    $v^*(k-1, j_1^{i,k-1}) + c_{ij_1^{i,k-1}} \leq \cdots \leq v^*(k-1, j_{|FS(i)|}^{i,k-1}) + c_{ij_{|FS(i)|}^{i,k-1}}$;
    Initialize $v^*(k,i) = -\infty$ and $\check{v}(k-1,i) = \infty$;
    for $m = 1, \ldots, \min\{k, |FS(i)|\}$ **do**
      if $\check{v}(k-m,i) > v^*(k-m, j_m^{i,k-m}) + c_{ij_m^{i,k-m}} + d_{ij_m^{i,k-m}}$ **then**
        $\check{v}(k-m,i) = v^*(k-m, j_m^{i,k-m}) + c_{ij_m^{i,k-m}} + d_{ij_m^{i,k-m}}$;
      end
      if $m < |FS(i)|$ **then**
        $u_{kim} = \min\left\{\check{v}(k-m,i), v^*(k-m, j_{m+1}^{i,k-m}) + c_{ij_{m+1}^{i,k-m}}\right\}$;
      end
      else
        $u_{kim} = \check{v}(k-m,i)$;
      end
      if $v^*(k,i) < u_{kim}$ **then**
        $v^*(k,i) = u_{kim}$;
      end
    end
  end
  Let Q be the set of temporarily labeled nodes, with $Q = \mathcal{N}$ initially;
  Initialize $u_{ki0} = \infty$, $\forall i \in \mathcal{N} \setminus \{t\}$, and $u_{kt0} = 0$;
  for $h = 1, \ldots, |\mathcal{N}|$ **do**
    Let j be a node having the minimum value of $\max\{v^*(k,j), u_{kj0}\}$ in Q;
    and remove j from Q;
    $v^*(k,j) = \max\{v^*(k,j), u_{kj0}\}$;
    for *nodes* $i : (i,j) \in \mathcal{A}$ **do**
      if $u_{ki0} > v^*(k,j) + c_{ij}$ **then**
        Set $u_{ki0} = v^*(k,j) + c_{ij}$;
      end
    end
  end
end
Return $v^*(b,s)$;

Algorithm 24: DSPI-UB(G, b)

Select a simple path $\mathcal{P}$ from s to t in G;
Transform G into a DAG, G', by eliminating arcs from G other than those in $\mathcal{P}$;
Compute z_{UB} as the output of DP-DSPI-DAG(G',b);

V. Papers Published

REFEREED JOURNAL PUBLICATIONS

Behdani, B., Yun, Y., Smith, J.C., and Xia, Y., "Decomposition Algorithms for Maximizing the Lifetime of Wireless Sensor Networks with Mobile Sinks," *Computers and Operations Research*, 39(5), 1054-1061, 2012.

Behdani, B., Smith, J.C., and Xia, Y., "The Lifetime Maximization Problem in Wireless Sensor Networks with a Mobile Sink: MIP Formulations and Algorithms, *IIE Transactions*, 45(10), 1094-1113, 2013.

Behdani, B. and Smith, J.C., "An Integer-Programming-Based Approach to the Close-Enough Traveling Salesman Problem," *INFORMS Journal on Computing*, 26(3), 415-432, 2014.

Buchanan, A., Walteros, J.L., Butenko, S., and Pardalos, P.M., "Solving Maximum Clique in Sparse Graphs: An $O(nm+ n2^{d/4})$ algorithm for d-degenerate graphs," *Optimization Letters*, 8(5), 1611-1617, 2014.

Buke, B., Smith, J.C., and Thomas, S.A., "On a Random Walk Reliability Problem with Arc Memory," *Networks*, to appear, 2015.

Buyuktahtakin, I.E., Smith, J.C., Hartman, J.C., and Luo, S., "Parallel Asset Replacement Problem under Economies of Scale with Multiple Challengers," *The Engineering Economist*, 59(4), 237-258, 2014.

Dinh, T.N., Nguyen, N.P., Alim, M.A., and Thai, M.D., "A Near-optimal Adaptive Algorithm for Maximizing Modularity in Dynamic Scale-free Networks," *Journal of Combinatorial Optimization*, 30(3), 747-767, 2015.

Dinh, T.N. and Thai, M.T., "Community Detection in Scale-free Networks: Approximation Algorithms for Maximizing Modularity," *IEEE Journal on Selected Areas in Communications*, 31(6), 997-1006, 2013.

Dinh, T.N. and Thai, M.T., "Towards Optimal Community Detection: From Trees to General Weighted Networks," *Internet Mathematics* 11(3), 181-200, 2015.

Dinh, T.N., Xuan, Y., Thai, M.T., Pardalos, P.M., and Znati, T., "On New Approaches of Assessing Network Vulnerability: Hardness and Approximation," *IEEE/ACM Transactions on Networking (ToN)*, 20(2), 609-619, 2012.

Fan, N., Pardalos, P. M., and Zheng, Q. P., "Robust Optimization of Graph Partitioning Involving Interval Uncertainty," *Theoretical Computer Science*, 447, 53-61, 2012.

Georgiev, P. G., Luc, D. T., and Pardalos, P. M., "Robust Aspects of Solutions in Deterministic Multiple Objective Linear Programming," *European Journal of*

Operational Research, 229(1), 29-36, 2013.

Granata, D., Behdani, B., and Pardalos, P. M., "On the Complexity of Path Problems in Properly Colored Directed Graphs," *Journal of Combinatorial Optimization,* 24, 459-467, 2012.

Hemmati, M. and Smith, J.C., "A Mixed-Integer Bilevel Programming Approach for a Competitive Prioritized Set Covering Problem," under revision for *Discrete Optimization*, 2015.

Hemmati, M., Smith, J.C., and Thai, M.T., "A Cutting-plane Algorithm for Solving a Weighted Influence Interdiction Problem," *Computational Optimization and Applications*, 57(1), 71-104, 2014.

Lozano, L. and Smith, J.C., "A Backward Sampling Framework for Interdiction Problems with Fortification", submitted to *INFORMS Journal on Computing*, 2015.

Mishra, S., Dinh, T. N., Thai, M. T., Seo, J., Shin, I., "Optimal Packet Scan Against Malicious Attacks in Smart Grids," *Theoretical Computer Science*, to appear, 2015.

Nguyen, N.P., Dinh, T.N., Shen, Y. and Thai, M.T., "Dynamic Social Community Detection and its Applications," *PLoS ONE* 9(4), e91431, 2014.

Nguyen, N.P., Alim, M.A., Dinh, T.N. and Thai, M.T., "A Method to Detect Communities with Stability in Social Networks," *Social Network Analysis and Mining* 4(1), 10.1007/s13278-014-0224-2, 2014.

Nguyen, D.T., Shen, Y., and Thai, M.T., "Detecting Critical Nodes in Interdependent Power Networks for Vulnerability Assessment," *IEEE Transactions on Smart Grid*, 4(1), 151-159, 2013.

Penuel, J., Smith, J.C., and Shen, S., "Models and Complexity Analysis for the Graph Decontamination Problem with Mobile Agents," *Networks*, 61(1), 1-19, 2013.

Prince, M., Geunes, J., and Smith, J.C., "Procurement Allocation Planning with Multiple Suppliers under Competition," *International Journal of Production Research*, 51(23-24), 6900-6922, 2013.

Prince, M. and Smith, J.C., "Capacitated Facility Interdiction Problems with Fortification under Median and Center Objective Functions," under revision for *Transportation Science*, 2015.

Prince, M., Smith, J.C., and Geunes, J., "A Three-Stage Procurement Optimization Problem Under Uncertainty," *Naval Research Logistics*, 60(5), 395-412, 2013.

Romich, A., Lan, G., and Smith, J.C., "Optimizing Placement of Stationary Monitors,"

IIE Transactions, 47(6), 556-576, 2015a.

Romich, A., Lan, G., and Smith, J.C., "A Robust Sensor Covering and Communication Problem", under revision for *Naval Research Logistics*, 2015b.

Sefair, J. and Smith, J.C., "Dynamic Shortest-Path Interdiction", submitted to *Networks*, 2015.

Shen, S. and Smith, J.C., "A Decomposition Approach for Solving a Broadcast Domination Network Design Problem," *Annals of Operations Research*, 210(1), 333-360, 2013.

Shen, S. and Smith, J.C., "Polynomial-time Algorithms for Solving a Class of Critical Node Problems on Trees and Series-Parallel Graphs," *Networks*, 60(2), 103-119, 2012.

Shen, S., Smith, J.C., and Goli, R., "Exact Interdiction Models and Algorithms for Disconnecting Networks via Node Deletions," *Discrete Optimization*, 9(3), 172-188, 2012.

Shen, Y., Nguyen, D.T., Xuan, Y., and Thai, M. T., "New Techniques for Approximating Optimal Substructure Problems in Power-Law Graphs," *Theoretical Computer Science*, 447, 107-119, 2012.

Sorokin, A, Boginski, V., Nahapetyan, N., and Pardalos, P.M., "Computational Risk Management Techniques for Fixed Charge Network Flow Problems with Uncertain Arc Failures," *Journal of Combinatorial Optimization*, 25, 99-122, 2013.

Sullivan, K.M., Morton, D.P., Pan, F., and Smith, J.C., "Securing a Border under Asymmetric Information," *Naval Research Logistics*, 61(2), 91-100, 2014.

Sullivan, K.M. and Smith, J.C., "Exact Algorithms for Solving a Euclidean Maximum Flow Network Interdiction Problem," *Networks*, 64(2), 109-124, 2014.

Sullivan, K.M., Smith, J.C., and Morton, D.P., "Convex Hull Representation of the Deterministic Bipartite Network Interdiction Problem," *Mathematical Programming*, 145(1-2), 349-376, 2014.

Syed, M. N., Georgiev, P. G., and Pardalos, P. G., "A Hierarchical Approach for Sparse Source Blind Signal Separation Problem," *Computers and Operations Research*, 41, 386-398, 2014.

Tadayon, B. and Smith, J.C., "Algorithms for an Integer Multicommodity Network Flow Problem with Node Reliability Considerations," *Journal of Optimization Theory and Applications*, 161(2), 506-532, 2014.

Tadayon, B. and Smith, J.C., "Algorithms and Complexity Analysis for Robust Single-Machine Scheduling Problems," *Journal of Scheduling*, to appear, 2015.

Tang, Y., Richard, J.-P., and Smith, J.C., "A Class of Algorithms for Mixed-Integer Bilevel Min-Max Optimization," *Journal of Global Optimization*, to appear, 2015.

Tiwari, T., Dinh, T. N., and Thai, M. T., "On Centralized and Localized Approximation Algorithms for Interference-Aware Broadcast Scheduling," *IEEE Transactions on Mobile Computing*, 12(2), 233-247, 2013.

Vogiatzis, C. and Pardalos, P.M., "Evacuation Modeling and Betweenness Centrality," *European Journal on Computational Optimization*, under review, 2015.

Vogiatzis, C., Pasiliao, E., and Pardalos, P.M., "Graph Partitions for the Multidimensional Assignment Problem," *Computational Optimization and Applications*, 58(1), 205-224, 2014.

Vogiatzis, C., Yoshida, R., Aviles-Spadoni, I., Imamato, S., and Pardalos, P.M., "Livestock Evacuation Planning for Natural and Man-made Emergencies," *International Journal of Mass Emergencies and Disasters*, 31(1), 25-37, 2013.

Walteros, J.L., Vogiatzis, C., Pasiliao, E.L., and Pardalos, P.M., "Integer Programming Models for the Multidimensional Assignment Problem with Star Costs," *European Journal of Operational Research*, 235(3), 553-568, 2014.

Xuan, Y., Shen, Y., and Nguyen, N. P., "Efficient Multi-Link Failure Localization in All-Optical Networks," *IEEE Transactions on Communications*, 61(3), 1144-1151, 2013.

Yun, Y., Xia, Y., Behdani, B., and Smith, J.C., "Distributed Algorithm for Lifetime Maximization in Delay-Tolerant Wireless Sensor Network with Mobile Sink," *IEEE Transactions on Mobile Computing*, 12(10), 1920-1930, 2013.

Zhang, H., Shen, Y., and Thai, M.T., "Robustness of Power-Law Networks: Its Assessment and Optimization," *Journal of Combinatorial Optimization*, to appear, 2015.

BOOKS

Goldengorin, B. and Pardalos, P. M., *Data Correcting Approaches in Combinatorial Optimization*, Springer, 2012.

Xanthopoulos, P., Pardalos, P. M., and Trafalis, T. B., *Robust Data Mining*, Springer, 2013.

EDITED BOOKS

Boginski, V., Commander, C., Pardalos, P.M., and Ye, Y., *Sensors: Theory, Algorithms, and Applications*, Springer, 2012.

Oliveira, C. and Pardalos, P.M., *Mathematical Aspects of Network Routing Optimization*, Springer, 2011.

Sorokin, A., Rebennack, S., Pardalos, P.M., Iliadis, N., and Pereira, M., *Handbook of Networks in Power Systems I*, Springer, 2011.

Sorokin, A., Rebennack, S., Pardalos, P.M., Iliadis, N., and Pereira, M., *Handbook of Networks in Power Systems II*, Springer, 2011.

Thai, M.T. and Pardalos, P.M., *Handbook of Optimization in Complex Networks: Communication and Social Networks*, Springer, 2011.

Thai, M.T. and Pardalos, P.M., *Handbook of Optimization in Complex Networks: Theory and Applications*, Springer, 2011.

Vogiatzis, C., Walteros, J.L., and Pardalos, P.M., *Dynamics of Information Systems: Computational and Mathematical Challenges*, Springer, 2014.

REFEREED BOOK CHAPTERS

Arulsevan, A., Commander, C.W., Shylo, O., Pardalos, P.M., "Cardinality-Constrained Critical Node Detection Problem," in *Performance Models and Risk Management in Communications Systems*, (N. Gülpınar, P. Harrison and B. Rüstem, editors), 46, 79-91, 2011.

Nguyen, N.P., Xuan, Y., and Thai, M.T., "On Detection of Community Structure in Dynamic Social Networks," in *Handbook of Optimization in Complex Networks: Communication and Social Networks*, (M. T. Thai and P. Pardalos, editors), Springer, 2011.

Pappu, V. and Pardalos, P.M., "High Dimensional Data Classification," in *Clusters, Orders, and Trees: Methods and Application* (F. Aleskerov, B. Goldengorin, and P.M. Pardalos, editors), 119-150, 2014.

Pardalos, P.M., Yatsenko, V.A., Fenn, M.B., and Chikrii, A.A., "Bilinear Markovian Processes of Search for Stationary and Moving Objects," in *Examining Robustness and Vulnerability of Networked Systems* (S. Butenko, E.L. Pasiliao, and V. Shylo, editors), 209-230, 2014.

Walteros J.L. and Pardalos, P.M., "Selected Topics in Critical Element Detection," in *Applications of Mathematics and Informatics in Military Science* (N. J. Daras, editor), 9-26, 2012.

REFEREED PROCEEDINGS

Alim, Md A., Kuhnle, A., and Thai, M.T., Are Communities As Strong As We Think?, Proceedings of IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining (ASONAM), 2014.

Alim, Md A., Nguyen, N.P., Dinh, T.N., and Thai, M.T., Vulnerability Analysis of Overlapping Communities in Complex Networks, in Proceedings of the 2014 IEEE/WIC/ACM International Conference on Web Intelligence (WI), 2014.

Buchanan, A., Walteros, J.L., Butenko, S., and Pardalos, P.M., "Solving Integer Programs with Dense Conflict Graphs," XII Global Optimization Workshop, 2014.

Dinh, T.N., Nguyen, N.P., and Thai, M.T., "An Adaptive Approximation Algorithm for Community Detection in Dynamic Scale-Free Networks," INFOCOM, 2013.

Dinh, T.N. and Thai, M.T., "Assessing Attack Vulnerability in Networks with Uncertainty," IEEE INFOCOM 2015.

Dinh, T.N. and Thai, M.T., "Finding Community Structures with Performance Guarantees in Complex Networks", Proceedings of IEEE International Conference on Social Computing (SocialCom), 2011.

Kuhnle, A., Li, X., and Thai, M.T., "Online Algorithms for Optimal Resource Management in Dynamic D2D Communications," Proceedings of the IEEE 10th International Conference on Mobile Ad-hoc and Sensor Networks (MSN), 130-137, 2014.

Mishra, S., Li, X., Kuhnle, A., Thai, M. T., and Seo, J., "Rate Alteration Attacks in Smart Grid," IEEE INFOCOM 2015.

Mishra, S., Li, X., Thai, M.T., and Seo, J., "Cascading Critical Nodes Detection with Load Redistribution in Complex Systems," Proceedings of the 8th Annual International Conference on Combinatorial Optimization and Applications (COCOA), 2014.

Nguyen, D.T., Alim, M.A., Shen, Y., and Thai, M.T., "Assessing Network Vulnerability in a Community Structure Point of View," Proceedings of IEEE/ACM Int Conf on Advances in Social Networks Analysis and Mining (ASONAM), 2013.

Nguyen, D.T., Das, S., and Thai, M.T., "Influence Maximization in Multiple Online Social Networks," Proceedings of the IEEE Global Communication Conference, 3060-3065, 2013.

Nguyen, D.T., Zhang, H., Das, S., Thai, M.T., and Dinh, T.N., "Least Cost Influence in Multiplex Social Networks," Proceedings of the IEEE International Conference on Data Mining, 567-576, 2013.

Nguyen, N.P., Dinh, T.N., Nguyen, D.T., and Thai, M.T., "Overlapping Community Structures and their Detection on Social Networks," Proceedings of IEEE International Conference on Social Computing (SocialCom), 2011.

Nguyen, N.P., Dinh, T.N., Tokala, S., and Thai, M.T., "Overlapping Communities in Dynamic Networks: Their Detection and Mobile Applications," Proceedings of ACM International Conference on Mobile Computing and Networking (MobiCom), 2011.

Nguyen, N.P., Dinh, T.N., Xuan, Y., and Thai, M.T., "Adaptive Algorithms for Detecting Community Structure in Dynamic Social Networks," Proceedings of the IEEE Communications Society (INFOCOM), 2011.

Nguyen, N.P. and Thai, M.T., "Finding Overlapped Communities in Online Social Networks with Nonnegative Matrix Factorization," in *Proceedings of the IEEE Military Communications Conference* (MILCOM), 2012.

Nguyen, N.P., Dinh, T.N., Xuan, Y., and Thai, M.T., "A Novel Method for Worm Containment on Dynamic Social Networks," Proceedings of the IEEE Military Communications Conference (MILCOM), 2010.

Pardalos, P.M., Resende, M.G.C., Vogiatzis, C., and Walteros, W.L., "Learning and Intelligent Optimization," 8th International Conference, LION 8, 2014.

Shen, Y., Dinh, T.N., and Thai, M.T., "Adaptive Algorithms for Detecting Critical Links and Nodes in Dynamic Networks," Proceedings of the IEEE Military Communications Conference (MILCOM), 2012.

Shen, Y., Li, X., and Thai, M.T., "Approximation Algorithms for Optimization Problems in Random Power-Law Graphs," Proceedings of the 8th Annual International Conference on Combinatorial Optimization and Applications (COCOA), 2014.

Shen, Y., Nguyen, D.T., and Thai, M.T., "Adaptive Approximation Algorithms for Hole Healing in Hybrid Wireless Sensor Networks," INFOCOM, 2013.

Shen, Y., Nguyen, N.P., and Thai, M.T., "Exploiting the Robustness on Power-Law Networks, in Proceedings of the 17th Int Computing and Combinatorics Conference (COCOON), 2011.

Shen, Y., Xuan, Y., and Thai, M.T., "On Local Approximation of Minimum-Latency Broadcast Scheduling in 3D MANETs," Proceedings of the IEEE Military Communications Conference (MILCOM), 2011.

Walteros, J. L. and Pardalos, P.M., “A Decomposition Approach for Solving Critical Clique Detection Problems,” Proceedings of the 10th International Symposium on Experimental Algorithms. Springer Lecture Notes in Computer Science, v7276, pp. 393-404, 2012

VI. Presentations (in addition to conference proceedings above)

Smith, J.C., "Tutorial: Teaching DP and duality via games, interdiction, and robust optimization," Invited Lecture, 10/20/12, INFORMS 2012 Conference, Phoenix, AZ.

Smith, J.C., "Revisiting fortification algorithms for facility interdiction problems," Invited Lecture, 11/15/12, Naval Postgraduate School, Monterey, CA.

Smith, J.C., "Defense algorithms for capacitated facility interdiction problems," Invited Lecture, 01/07/13, INFORMS Computing Society Conference, Santa Fe, NM.

Pardalos, P.M., "Detecting critical subsets in networks," Invited Lecture, 02/08/13, Mechanical and Industrial Engineering Seminar, Toronto.

Pardalos, P.M., "Assessing the vulnerability of evacuation plans via critical element detection," Invited Lecture, 02/14/13, AAAS Annual Meeting, Boston, MA.

Pardalos, P.M., "Optimization and modeling in energy systems," Keynote Lecture, 03/23/13, Systems and Optimization Aspects of Smart Grid Challenges, Tucson, AZ.

Smith, J.C., "A robust sensor covering and communication problem," Invited Lecture, 05/24/13, Industrial and Systems Engineering Research Conference, San Juan, PR.

Pardalos, P.M., "Networks everywhere," Invited Lecture, 06/03/13, NATO advanced research workshop on examining robustness and vulnerability of critical infrastructure networks, Kyiv, Ukraine.

Pardalos, P.M., "Global optimality conditions in non-convex optimization and related issues," Keynote Lecture, 06/19/13, Numerical Computations: Theory and Algorithms, Calabria, Italy.

Pardalos, P.M., "Cliques and quasi-cliques in large graphs: Theory and applications," Keynote Lecture, 06/24/13, Discrete Optimization and Operations Research, Novosibirsk, Russia.

Pardalos, P.M., "General Optimization Discussion," Invited Lecture, 07/10/13, International Conference on OR for Development, Lleida, Spain.

Thai, M.T., "Least Cost Influence in Multiplex Social Networks: Model Representation and Analysis," Invited Lecture, 12/01/13, IEEE, Dallas, TX.

Thai, M.T., "Influence Maximization in Multiple Online Social Networks," Invited Lecture, 12/10/13, IEEE, Atlanta, GA.

Smith, J.C., "Revisiting Fortification Algorithms for Facility Interdiction Problems," Invited Lecture, 02/01/14, University of Buffalo, Buffalo, NY.

Smith, J.C., "Revisiting Fortification Algorithms for Facility Interdiction Problems," Invited Lecture, 03/01/14, Southern California University, Los Angeles, CA.

Smith, J.C., "Revisiting Fortification Algorithms for Facility Interdiction Problems," Invited Lecture, 04/01/14, University of Tennessee, Knoxville, TN.

Sefair, J., "Dynamic Shortest-Path Interdiction ," Invited Lecture, 11/11/14, INFORMS Annual Meeting, San Francisco, CA.

Lozano, L., "A backward sampling framework for interdiction problems with fortification," Invited Lecture, 11/11/14, INFORMS Annual Meeting, San Francisco, CA.

Thai, M.T., "Online Algorithms for Optimal Resource Management in Dynamic D2D Communications," Invited Lecture, 12/19/14, IEEE 10th International Conference on Mobile Ad-hoc and Sensor Networks (MSN), Maui, HI.

Thai, M.T., "Heterogenous Interdependent Networks: Critical Elements and Cascades Analysis," Keynote Lecture, 12/19/14, 8th Annual International Conference on Combinatorial Optimization and Applications (COCOA), Maui, HI.

Thai, M.T., "Cascading Critical Nodes Detection with Load Redistribution in Complex Systems," Invited Lecture, 12/20/14, 8th Annual International Conference on Combinatorial Optimization and Applications (COCOA), Maui, HI.

Thai, M.T., "Approximation Algorithms for Optimization Problems in Random Power-Law Graphs," Invited Lecture, 12/20/14, 8th Annual International Conference on Combinatorial Optimization and Applications (COCOA), Maui, HI.

Walteros, J.L., "A mathematical framework for detecting Sybil communities in online social networks," Invited Lecture, 01/11/15, INFORMS Computing Society Conference, Richmond, VA.

Smith, J.C., "Dynamic Shortest Path Interdiction," Invited Lecture, 01/11/15, INFORMS Computing Society Conference, Richmond, VA.

Vogiatzis, C., "Assessing the Vulnerability of Evacuation Plans via Critical Element Detection," Invited Lecture, 03/14/15, 56th Annual Transportation Research Forum, Atlanta, GA.

Pardalos, P.M., "Cricitcal element selection in large networks," Invited Lecture, 04/26/15, Seminar Series: Anhui University of Finance and Economics, Bengbu, China.

Pardalos, P.M., "Clustering and search techniques in networks," Invited Lecture, 04/30/15, Seminar Series: Hefei University of Technology, Hefei, China.

Thai, M.T., "Rate Alteration Attacks in Smart Grid," Invited Lecture, 04/30/15, IEEE INFOCOM 2015, Hong Kong, China.

Dang, T.N., "Assessing Attack Vulnerability in Networks with Uncertainty," Invited Lecture, 04/30/15, IEEE INFOCOM 2015, Hong Kong, China.

Smith, J.C., "A backward sampling framework for interdiction problems with fortification," Invited Lecture, 05/04/15, Seminar Series: Bogazici University, Istanbul, Turkey.

Smith, J.C., "A backward sampling framework for interdiction problems with fortification," Invited Lecture, 05/05/15, Seminar Series: Sabanci University, Istanbul, Turkey.

Smith, J.C., "A backward sampling framework for interdiction problems with fortification," Invited Lecture, 05/06/15, Seminar Series: Koc University, Istanbul, Turkey.

Curry, R., "A Semi-Continuous Formulation to Maximize Wireless Sensor Network Lifetime," Invited Lecture, 05/31/15, Industrial and Systems Engineering Research Conference, Nashville, TN.

Lozano, L., "A backward sampling framework for interdiction problems with fortification," Invited Lecture, 05/31/15, Industrial and Systems Engineering Research Conference, Nashville, TN.

Sefair, J., "Dynamic Shortest-Path Interdiction Games," Invited Lecture, 06/02/15, Industrial and Systems Engineering Research Conference, Nashville, TN.

Pardalos, P.M., "Evacuation modeling and betweenness centrality," Invited Lecture, 06/29/15, Dynamics of Disasters 2015, Kalamata, Greece.

VII. Students Graduated with DTRA Support

Academic Affiliations are listed here if applicable.

Ph.D. student Behnam Behdani graduated in 2012.

Ph.D. student Thang Dinh graduated in 2013 and now serves as an assistant professor at Virginia Commonwealth University.

Ph.D. student Neng Fan graduated in 2011 and now serves as an assistant professor at the University of Arizona.

Ph.D. student Vijay Pappu graduated in 2013.

Ph.D. student Dung Nguyen graduated in 2014.

Ph.D. student Nam Nguyen graduated in 2013 and now serves as an assistant professor at Towson University.

Ph.D. student Michael Prince graduated in 2013.

Ph.D. student Jorge Sefair defended his dissertation in June 2015 and will begin an academic career as a (tenure-track) assistant professor at Arizona State University in August 2015.

Ph.D. student Siqian Shen graduated in 2011 and now serves as an assistant professor at the University of Michigan.

Ph.D. student Siqian Shen graduated in 2013.

Ph.D. student Sibel Sonuc graduated in 2012 and now serves as a postdoctoral assistant at Northeastern University.

Ph.D. student Kelly Sullivan graduated in 2012 and now serves as an assistant professor at the University of Arkansas.

Ph.D. student Yu-Song Syu graduated in 2012.

Ph.D. student Bita Tadayon graduated in 2014.

Ph.D. student Yen Tang graduated in 2013.

Ph.D. student Chrysafis Vogiatzis graduated in 2014 and will begin an academic career as a (tenure-track) assistant professor at North Dakota State University in August 2015.

Ph.D. student Jose Walteros graduated in 2014 and now serves as an assistant professor at the University at Buffalo.

Ph.D. student Hongsheng Xu graduated in 2012.

Current Ph.D. students who have received DTRA support: Md Abdul Alim, Robert Curry, Xiang Li, Subhankar Mishra, Ioannis Pappas, Jiaxing Pi, Huiling Zhang, Huiyuan Zhang

VIII. Impacts

Dr. Pardalos was named the first Paul and Heidi Brown Preeminent Professor of Industrial and Systems Engineering at the University of Florida.

Dr. Smith was named Professor and Chair of the Industrial Engineering department at Clemson University.

Dr. Pardalos won the following awards for his contributions to the field of operations research during the award period:

- "Numerical Computation: Theory and Algorithms" conference award for contributions to optimization
- Medal in honor of broad contributions to science and engineering, University of Catania
- EURO Gold Medal Award

Dr. Thai won the following awards due to her DTRA-sponsored work.

- Provost's Excellence Award for Associate Professors at the University of Florida, 2010.
- Early promotion to Full Professor in 2015.
- Her student, Thang Dinh, received UFIC Outstanding International Student Award.
- Best conference paper award for "Online Algorithms for Optimal Resource Management in Dynamic D2D Communications," in *Proceedings of the IEEE 10th International Conference on Mobile Ad-hoc and Sensor Networks* (MSN).

Sullivan, K.M. and Smith, J.C., "Exact Algorithms for Solving a Euclidean Maximum Flow Network Interdiction Problem," Networks, 64(2), 2014 won the Glover-Klingman Prize for best paper in Networks.

Romich, A., Lan, G., and Smith, J.C., "Optimizing Placement of Stationary Monitors," IIE Transactions, 47(6), 556-576, 2015 was a featured article in the *Industrial Engineering* magazine.

DTRA-research-related topics were incorporated into five Ph.D.-level classes at the University of Florida:

- Advanced Network Optimization
- Linear Programming Extensions
- Optimization in Adaptive Complex Systems and Social Networks
- Approximation Algorithms

- Data Mining

The latter class was prepared by Co-PI Pardalos, who taught this as an honors level course as a part of his Dunlevie Professorship award, which he won during the 2013-14 academic year.

IX. Tech Transition

The PI for the DTRA grant was awarded a grant by the *Office of Naval Research* (ONR), running from January 2013 through the end of December 2015, The title of the ONR grant is, "Dynamic and Adaptive Sensor Operations Under Uncertainty," and was awarded to Dr. Smith and to a Co-PI, Dr. Guanghui Lan. The grant is funded under the ONR program 12-SN-0009, and is under the 6.2 funding category.

The subject of our DTRA grant examines cascading failures in defender-attacker-defender settings. For instance, a problem we would investigate in the DTRA proposal might examine a situation in which an attacker destroys links in order to maximize the shortest possible path a defender could take in some network. The objective would then be to fortify links first, so that the attacker's objective is as small as possible (i.e., the defender minimizes the maximum possible shortest path length induced by the attacker). Of particular interest are those problems in which the network is multi-layered and complex.

As a result of the theoretical and methodological advances made in the DTRA grant, we were able to explore sensor operations arising in the defender-attacker-defender setting, for the ONR proposal. An example of the type of problem considered in the ONR program would be similar to the one described above, but in which the "attacker" is not necessarily an adversary, and the "defender" is not necessarily the protagonist. Hence, it makes more sense to refer to these agents as "leader" and "follower" to better fit the problem context. For instance, suppose that the follower is a terrorist that seeks to gain access to sensitive infrastructure. The leader would place sensors (or monitors) in a continuous region in order to minimize the maximum probability with which the follower could reach the infrastructure undetected. This problem was inspired by the defense of a nuclear power plant with access to public waterways, which might be exploited by a terrorist.

In another setting, a set of targets may need to be covered by sensors. The sensors also must communicate with one another wirelessly, suffering some degradation that occurs as a function of distance between the sensors. Complicating matters is the observation that once sensors are placed, they might drift by some vector. Drifting could occur due to placement error, or in the case of unmanned (aerial or underwater) vehicles, due to physical currents. To model the robustness of our sensor placement, we bound placement error by a parameter, and conceptually envision that an adversary moves each sensor by a vector of length no more than

this parameter. Hence, the game proceeds as follows (in the defender-attacker-defender language): The defender first places its sensors, the attacker perturbs the sensors, and then the defender performs optimal communication among each pair of sensors in their adjusted locations. This is a max-min-max model, where the inner minimization problem represents the realization of a worst-case-scenario (which is defined relative to each possible first-stage sensor location decision) rather than an actual malicious entity.

DEPARTMENT OF DEFENSE

DEFENSE THREAT REDUCTION
AGENCY
8725 JOHN J. KINGMAN ROAD
STOP 6201
FORT BELVOIR, VA 22060
ATTN: P. VANDEVENTER

DEFENSE TECHNICAL
INFORMATION CENTER
8725 JOHN J. KINGMAN ROAD,
SUITE 0944
FT. BELVOIR, VA 22060-6201
ATTN: DTIC/OCA

DEPARTMENT OF DEFENSE CONTRACTORS

QUANTERION SOLUTIONS, INC.
1680 TEXAS STREET, SE
KIRTLAND AFB, NM 87117-5669
ATTN: DTRIAC